DC-Text - a simple text-based format for DC metadata

Pete Johnston
Eduserv Foundation
Tel: +44 1225 474323
pete.johnston@eduserv.org.uk

Andy Powell
Eduserv Foundation
Tel: +44 1225 474319
andy.powell@eduserv.org.uk

Abstract

All development activities undertaken by the Dublin Core Metadata Initiative (DCMI) are now underpinned by the DCMI Abstract Model (DCAM), the syntax-neutral framework within which Dublin Core (DC) metadata descriptions are constructed. As such, it is vital that the DCMI community has a simple way of articulating the DCAM and of sharing example metadata descriptions based on it. This paper describes a simple format for representing DC metadata descriptions in plain text, known as "DC-Text". The format is intended to be useful as a means of representing DC metadata examples in human-readable documents in a way that highlights the constructs of the DCAM.

Keywords:
Dublin Core, syntax, abstract model.

1. Introduction

The DCAM [1] describes the entities that make up DC metadata *description sets*. This paper describes a simple format for representing such *description sets* in plain text, known as "DC-Text".

Although rigorously defined (for example, a BNF grammar is available) the primary motivation for developing the DC-Text format was not to create a new machine to machine interchange format comparable with the other encoding syntaxes provided by DCMI. Rather, DC-Text is primarily intended as a human-readable format, to be used to facilitate discussion between people in the DCMI community - particularly those people that are involved in developing and deploying DC-based solutions and standards.

2. DCMI Abstract Model

The DCAM describes the entities that make up DC metadata *description sets*.

According to the DCAM:

- a *description set* is made up of one or more *descriptions*
- a *description* is made up of
  - zero or one *resource URI* and
  - one or more *statements*
- a *statement* is made up of
  - exactly one *property URI* and
  - zero or one reference to a value in the form of a *value URI*
  - zero or more representations of a value, each in the form of a *value*

    representation
    - zero or one *vocabulary encoding scheme URI*
- a *value representation* is either
    - a *value string* or
    - a *rich representation*
- a *value string* may have an associated *value string language*
- a *value string* may have an associated *syntax encoding scheme URI*
- each *value* may be the subject of a *related description*

Description sets are encoded using one of the available encoding syntaxes to form *records*, digital objects that can be shared between software applications (typically across the network) in order to facilitate the exchange of information between services.

3. Encoding Syntaxes

At the time of writing, DCMI provides three encoding syntaxes [2] – (X)HTML meta tags, an XML format and RDF/XML.

These three encoding syntaxes were developed at different stages of the life of the DCMI. It is worth noting that all of them pre-date the DCAM and that they all use slightly different terminology. Although they are based on "abstract models" for DC metadata, those models differ from the DCMI Abstract Model in some significant respects. As a result, interpreting the existing encoding syntaxes from the perspective of the DCAM is problematic.

Together with the relative newness of the DCAM, this can lead to some confusion when the use of the DCAM is discussed by members of the DCMI community. This confusion is particularly notable where new DC application profiles are being developed, since a very solid understanding of the DCAM is required in this context.

Although the original purpose of the DCAM was as a reference against which different encoding syntaxes for DC metadata could be compared, its use now underpins all developments undertaken by the DCMI. As such, it is critical for the community to have ways of talking about the DCAM, and ways of representing example *descriptions sets* that make it easy for people to create and understand those examples and that is appropriate for use in the discussion forums typically used by the DCMI community - notably email discussion lists and Wikis.

The three existing encoding syntaxes do not fulfill this need, both because they are relatively complex for people (as opposed to machines) to create and read and, perhaps more importantly, because they do not offer a consistent way of talking about all aspects of the DCAM.

4. DC-Text

As a solution to this problem, DC-Text provides a simple format for representing a DC metadata *description set* that:

- is easy for people to use, both to create examples and to understand the examples provided by others,
- is suitable for use in online discussion forums such as email discussion lists and Wikis, and
- supports all aspects of the DCAM.

The format is also largely independent of specifications such as XML and Namespaces in XML, which minimizes the potential for confusion between features of the DCAM and features of XML.

The design of DC-Text was significantly influenced by work on text-based syntaxes within the Semantic Web community, where there has been a similar interest in developing formats that represent the "abstract models" being used while also remaining relatively simple and easy for humans to read and write. It was influenced in particular by Turtle [3] and OWL Web Ontology Language Concrete Abstract Syntax [4].

The general structure of a DC-Text document is as follows:

```
namespace declaration

label (
  label ( content )
  label (
    label ( [...] )
    [ ... ]
  )
)
```

Each of the primary entities in a DC metadata *description set*, as defined by the DCAM, is represented in DC-Text by a syntactic structure of the form:

```
label ( content )
```

where `label` is replaced by one of the following strings:

```
DescriptionSet, Description, DescriptionId, ResourceURI, Statement, PropertyURI,
DescriptionRef, VocabularyEncodingSchemeURI, ValueURI, ValueString, Language,
SyntaxEncodingSchemeURI, RichRepresentation, RepresentationURI, Base64
```

and `content` is either:

- a sequence of one or more syntactic structures of the form `label( content )` (i.e. these structures are "nested"); or
- a string of the form `"literal"`, which represents that Unicode literal; or
- a string of the form `<uri>`, which represents a URI; or
- a string of the form `prefix:name`, which represents a "qualified name" used as an abbreviation for a URI
- a string which represents a language tag (e.g. `en-GB`)
- a string which is a locally-scoped identifier used to establish relationships between *values* and their *descriptions*

For each type of `label` in the list above, the permitted form of `content` is determined by the syntax rules specified in Appendix A of the draft DC-Text specification [5].

DC-Text supports the representation of only a single *description set*. Therefore, each DC-Text representation consists of zero or more `namespace declarations` followed by a single `label( content )` structure with a `label` of `DescriptionSet`, and as `content`, one or more nested `label( content )` structures with a `label` of `Description`.

In summary, every DC-Text representation has the following outline form:

```
@prefix prefix: <uri> .

DescriptionSet (
  Description (
    Statement ( ... )
    Statement ( ... )
  )
  Description (
    Statement ( ... )
    Statement ( ... )
  )
)
```

Readers are referred to the draft DC-Text specification for the full description of the format, including the full BNF grammar. However, the two examples provided in the appendix to this paper are intended to show use of the DC-Text format in what is hopefully a relatively intuitive form without the need for a full syntactic explanation.

References:

1. A. Powell, M. Nilsson, A. Naeve and P. Johnston. DCMI Abstract Model. DCMI, 2005. <http://dublincore.org/documents/abstract-model/>
2. DCMI Encoding Guidelines. <http://dublincore.org/resources/expressions/>
3. D. Beckett. Turtle - Terse RDF Triple Language. 2006. <http://www.dajobe.org/2004/01/turtle/>
4. S. Bechhofer, P. Patel-Schneider and D. Turi. OWL Web Ontology Language Concrete Abstract Syntax. 2003. <http://owl.man.ac.uk/2003/concrete/latest/>
5. P. Johnston. DC-Text: A text syntax for Dublin Core metadata. 2006. <http://dublincore.org/architecturewiki/DCText>

Appendix

Example 1 - Simple DC e-print description

This example shows a simple DC *description* for one of the papers presented at the DC-2005 conference in Madrid:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .

DescriptionSet (
  Description (
    Statement (
      PropertyURI ( dc:title )
      ValueString ( "SKOS and the Ontogenesis of Vocabularies" )
    )
    Statement (
      PropertyURI ( dc:creator )
      ValueString ( "Joseph T. Tennis" )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( "controlled vocabularies" )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( "SKOS" )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( "versioning" )
    )
    Statement (
      PropertyURI ( dc:identifier )
      ValueString ( "http://purl.org/dcpapers/2005/Paper33" )
    )
  )
)
```

Example 2 - Qualified DC e-print description

This example shows a qualified DC *description set* for the paper used in example 1, this time providing one of the keywords in two languages and using multiple *descriptions* to provide additional information about the author and to explicitly separate information about the e-print 'work' and the e-print 'manifestation':

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms <http://purl.org/dc/terms/> .
@prefix foaf <http://xmlns.com/foaf/0.1/> .

DescriptionSet (
  Description (
    Statement (
      PropertyURI ( dc:title )
      ValueString ( “SKOS and the Ontogenesis of Vocabularies” )
    )
    Statement (
      PropertyURI ( dc:creator )
      ValueURI ( <http://purl.org/net/joetennis> )
      ValueString ( “Joseph T. Tennis” )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( “controlled vocabularies”
        Language ( en )
      )
      ValueString ( “vocabulaires commandés”
        Language ( fr )
      )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( “SKOS” )
    )
    Statement (
      PropertyURI ( dc:subject )
      ValueString ( “versioning” )
    )
    Statement (
      PropertyURI ( dc:identifier )
      ValueString ( “http://purl.org/dcpapers/2005/Paper33” )
    )
    Statement (
      PropertyURI (dcterms:hasFormat )
      ValueURI ( <http://www.slais.ubc.ca/PEOPLE/faculty/tennis-p/dcpapers/paper33.pdf> )
    )
  )
  Description (
    ResourceURI ( <http://purl.org/net/joetennis> )
    Statement (
      PropertyURI ( foaf:name )
      ValueString ( “Joseph T. Tennis” )
    )
    Statement (
      PropertyURI ( foaf:mbox )
      ValueURI ( <mailto:jtennis@interchange.ubc.ca> )
    )
  )
  Description (
    ResourceURI ( <http://www.slais.ubc.ca/PEOPLE/faculty/tennis-p/dcpapers/paper33.pdf> )
    Statement (
      PropertyURI ( dc:format )
      VocabularyEncodingSchemeURI ( dcterms:IMT )
      ValueString ( “application/pdf” )
    )
    Statement (
      PropertyURI ( dcterms:modified )
      ValueString ( “2005-10-14”
        SyntaxEncodingSchemeURI ( dcterms:W3CDTF )
      )
    )
  )
)
```