Encoding DC in HTML, XML and RDF

Tutorial 1: Basic Syntax at DC-2005, Madrid
12 September 2005

Andy Powell
a.powell@ukoln.ac.uk
UKOLN, University of Bath, UK
http://www.ukoln.ac.uk/

UKOLN is supported by:
About me…

• Andy Powell
• UKOLN, University of Bath, UK
  • UKOLN is a ‘centre of expertise in digital information management for the UK’
• member of the DC Usage Board
• chair of the DC Architecture Working Group
About you…

- How many of you are librarians?

- How many of you are software developers (computer programmers)

- How many of you have created a Dublin Core description in HTML (or XML or RDF/XML)?
Contents

• an abstract model for DC (30 mins)
• encoding DC in XHTML (15 mins)
• encoding DC in XML (15 mins)
• encoding DC in RDF/XML (5 mins)
• practical examples
  • OAI Protocol for Metadata Harvesting and RSS (20 mins)
Important DCMI documents...

- DCMI Abstract Model
  http://dublincore.org/documents/abstract-model/
- Expressing Dublin Core in HTML/XHTML meta and link elements
  http://dublincore.org/documents/dcq-html/
- Guidelines for implementing Dublin Core in XML
  http://dublincore.org/documents/dc-xml-guidelines/
- Expressing Simple Dublin Core in RDF/XML
  http://dublincore.org/documents/dcmes-xml/
- Expressing Qualified Dublin Core in RDF/XML
  http://dublincore.org/documents/dcq-rdf-xml/
- Namespace Policy for the DCMI
  http://dublincore.org/documents/dcmi-namespace/
- DCMI Metadata Terms
  http://dublincore.org/documents/dcmi-terms/
Implementing DC

• this tutorial is about the mechanics of implementing DC in HTML, XML and RDF
• it doesn’t really consider which implementation strategy is the best!
• ask yourself two questions…
  • what am I trying to achieve?
  • does using HTML, XML or RDF help me achieve it?
• do software and services exist that will support the creation and use of my metadata?
DCMI abstract model
Why an abstract model?

- the first part of this tutorial isn’t going to show any syntax!
- why?
- because before we start creating DCMI descriptions we need to understand what kinds of things we want to be able to say about ‘resources’
- known as the DCMI abstract model
- note: a very simplified view of the model is presented here
What is a resource?

• W3C/IETF definition of resource is
  “…anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources.”

• i.e. a resource is “anything”
  • physical things (books, cars, people)
  • digital things (Web pages, digital images)
  • conceptual things (colours, points in time, subjects)
**DC and resources**

- but… this seems to be too wide for the things we can describe with DC!
  - can we really describe people using DC?
  - do people have titles and subjects?
- no… in general we only use DC to describe a sub-set of all *resources*
- anything covered by the DCMIType list…
  - Collection, Dataset, Event, Image (Still or Moving), Interactive Resource, Service, Software, Sound, Text, Physical Object
DCMI abstract model

• a description is made up of
  • one or more statements (about one, and only one, resource) and
  • optionally, the URI of the resource being described (resource URI)

• each statement is made up of
  • a property URI (that identifies a property)
  • a value URI (that identifies a value) and/or one or more representations of the value (value representations)
Value strings

• each value representation may take the form of a value string, a rich value or a related description

• note: not going to discuss rich values and related descriptions in this tutorial

• each value string is a simple, human-readable string that represents the resource that is the value of the property

• each value string may have an associated value string language that is an ISO language tag (e.g. en-GB)
Elements and refinements

- within DCMI, we often use the phrases ‘element’ and ‘element refinement’
- an ‘element’ is just another word for a property
- an ‘element refinement’ is a special kind of property (a sub-property) that shares some meaning with one other property but has narrower semantics
  - e.g. if “Ben is the illustrator of a Book” then it is also true to say that “Ben is a contributor to the Book”
Encoding schemes

• values and value strings can be ‘qualified’ by using encoding schemes

• a vocabulary encoding scheme is used to indicate the class of the value
  • e.g. the value is taken from LCSH

• a syntax encoding scheme is used to indicate how the value string is structured
  • e.g. the value string is a date structured according to the W3CDTF rules (“2004-10-12”)
The 1:1 principle

• notice that the model indicates that each description describes one, and only one, resource
• this is commonly referred to as the 1:1 principle
• however…
Description sets

• real-world metadata applications tend to be based on loosely grouped sets of descriptions (where the described resources are typically related in some way)

• known in the abstract model as description sets

• for example, a description set might comprise descriptions of both a painting and the artist
Records

- *description sets* are instantiated, for the purposes of exchange between software applications, in the form of metadata *records*
- each *record* conforms to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)

```xml
<dc:title>a document</dc:title>
<dc:creator>andy powell</dc:creator>
```
Simple vs. qualified DC?

- within DCMI, we often use the phrases “simple DC” and “qualified DC”
- “simple DC” only supports a single description using the 15 DCMES elements with value strings
- “qualified DC” supports all the features of the abstract model, and allows the use of all DCMI terms as well as other, non-DCMI, terms

note that not everyone agrees with my definitions!
Dumb-down

- the process of translating qualified DC into simple DC is normally referred to as ‘dumbing-down’

<table>
<thead>
<tr>
<th>uninformed</th>
<th>informed</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>value</td>
</tr>
<tr>
<td>ignore any <em>property</em> that isn't in the Dublin Core Metadata Element Set</td>
<td>use <em>value URI</em> (if present) or <em>value string</em> as new <em>value string</em></td>
</tr>
<tr>
<td>recursively resolve sub-property relationships until one of the 15 properties in the DCMES is reached, otherwise ignore</td>
<td>use knowledge of <em>rich values</em>, <em>related descriptions</em> or the <em>value string</em> and the <em>syntax encoding scheme</em> to create a new <em>value string</em></td>
</tr>
</tbody>
</table>
Model summary

record (encoded as XHTML, XML or RDF/XML)

description set

description (about a resource (URI))

statement

property (URI)

vocabulary encoding scheme (URI)

value (URI)

representation

value string

OR rich value

OR related description

syntax encoding scheme (URI)

language (e.g. en-GB)
Encoding DC in XHTML (and HTML!)
What is being described?

- a DC description embedded in an (X)HTML document describes that document
- if you want to describe something else, don’t embed it in the (X)HTML document!

...not everyone would agree with this...
The basics

- the DC description is embedded into the <head> section of the (X)HTML document

```html
<html>
<head>
  ..DC description goes here...
</head>
<body>
...
```
DCMES elements

• use the ‘name’ and ‘content’ attributes of the XHTML `<meta>` element to encode the DC element (one of the 15 DCMES elements) and its value string. Use the following pattern:

`<meta name="DC.element" content="Value string" />`

• for example:

`<meta name="DC.date" content="2001-07-18" />`

...the element names of the 15 DCMES elements always have a lower-case first letter...
Value strings

• *value strings* go in the XHTML `<meta>` element ‘content’ attribute…
• the string in the ‘content’ attribute is defined to be CDATA, i.e. a sequence of characters from the document character set which may include character entities

...long value strings may be wrapped across multiple lines as necessary...

...will need to escape some characters, `&`, `&lt;`, `&gt;`, etc…
Value string language

• where the language of the value string is indicated, it should be encoded using the ‘xml:lang’ attribute of the XHTML `<meta>` element. For example:

```xml
<meta name="DC.subject" xml:lang="en" content="seafood" />
<meta name="DC.subject" xml:lang="fr" content="fruits de mer" />
```
Repeated elements

- multiple property values should be encoded by repeating the XHTML `<meta>` element for that property, for example:

```html
<meta name="DC.title" content="First title" />
<meta name="DC.title" content="Second title" />
```
Other DC elements

- DC also has elements that are not part of the DCMES (the original 15), e.g. Audience
- use the same pattern but with a ‘DCTERMS’ prefix:

  `<meta name="DCTERMS.element" content="Value" />

- for example:

  `<meta name="DCTERMS.audience" content="software developers" />

...element names may be mixed-case but should always have a lower-case first letter...
**Element refinements**

- use the same pattern for element refinements:

  ```xml
  <meta name="DCTERMS.elementRefinement" content="Value" />
  ```

- for example:

  ```xml
  <meta name="DCTERMS.modified" content="2001-07-18" />
  ```
Encoding schemes

- encoding schemes are encoded using the 'scheme' attribute of the XHTML `<meta>` element, using the following pattern:

```
<meta name="DC.element"
    scheme="DCTERMS.Scheme"
    content="Value" />
```

- for example:

```
<meta name="DC.date"
    scheme="DCTERMS.W3CDTF"
    content="2001-07-18" />
```
The case of names

- *elements, element refinements and encoding schemes* should use the names specified in

DCMI Metadata Terms
http://dublincore.org/documents/dcmi-terms/
The case of names (2)

- element and element refinement names may be mixed-case but should always have a lower-case first letter
- encoding scheme names may be mixed-case but should always start with an upper-case letter

<meta name="DCTERMS.temporal"
    scheme="DCTERMS.Period" content="name=The Great Depression; start=1929; end=1939;"/>

Handling namespaces...

- the ‘DC.’ and ‘DCTERMS.’ prefixes are used to indicate the namespace from which the property is taken
- put the namespace URI in an XHTML `<link>` element:

```xml
<link rel="schema.DC" href="http://purl.org/dc/elements/1.1/"/>
<link rel="schema.DCTERMS" href="http://purl.org/dc/terms/"/>
```

- while any string is allowable as the prefix, current practice is to use ‘DC.’ and ‘DCTERMS.’
Value URIs

• where the value of a property is the URI of another resource (e.g. DC.relation) an alternative form of encoding using the XHTML `<link>` element is preferred. Use the following pattern:

```
<link rel="propertyName" href="valueURI" />
```

• for example:

```
<link rel="DC.relation" href="http://www.example.org/" />
<link rel="DCTERMS.references" href="http://www.example.org/176459.pdf" />
```
Mixing DC and non-DC

- DC metadata can be mixed with non-DC metadata in XHTML `<meta>` elements
- the following example embeds DC, AGLS and unspecified metadata properties in the same XHTML Web page:

```html
<link rel="schema.DC" href="http://purl.org/dc/elements/1.1/" />
<link rel="schema.AGLS" href="http://www.naa.gov.au/recordkeeping/gov_online/agls/1.2" />
<meta name="DC.title" content="Services to Government" />
<meta name="keywords" content="archives, information management, public administration" />
<meta name="AGLS.Function" scheme="AGIFT" content="recordkeeping standards" />
```
A couple of examples

• Simple DC
  example 1

• Qualified DC
  example 2

• ScreenCam of using DC-dot
  http://www.ukoln.ac.uk/metadata/dcdot/
Encoding DC in XML
Properties and values

• encode properties as XML elements and value strings as the content of those elements

• the name of the XML element should be an XML qualified name (QName) of the property

\[
<\text{dc:title}>\text{Dublin Core in XML}</\text{dc:title}>
\]

• do not use constructs like

\[
<\text{dc:title value="\text{Dublin Core in XML}"} />
\]
DCMES property names

- the *property* names for the 15 DCMES elements should be all lower-case

  `<dc:title>Dublin Core in XML</dc:title>`

- do not use

  `<dc:Title>Dublin Core in XML</dc:Title>`
Repeating properties

• multiple value strings should be encoded by repeating the XML element for that property

<dc:title>First title</dc:title>
<dc:title>Second title</dc:title>
Value string language

• where the language of the value is indicated, it should be encoded using the ‘xml:lang’ attribute

    <dc:subject xml:lang="en">seafood</dc:subject>
    <dc:subject xml:lang="fr">fruits de mer</dc:subject>
Container elements

• note that it is anticipated that records will be encoded within one or more container XML element(s) of some kind
• this tutorial makes no recommendations for the name of any container element, nor for the namespace that the element should be taken from
• candidate container element names include `<dc>`, `<dublinCore>`, `<resource>`, `<record>` and `<metadata>`
Simple DC example

• example 3
**Element refinements**

- *element refinements* should be treated in the same way as other *properties*
- for example:

```
<dcterms:available>2002-06</dcterms:available>
```

- do not use any of the following:

```
<dc:date refinement="available">2002-06</dc:date>
<dc:date type="available">2002-06</dc:date>
<dc:date>
  <dcterms:available>2002-06</dcterms:available>
</dc:date>
```
Encoding schemes

- *encoding schemes* should be implemented using the 'xsi:type' attribute of the XML element for the *property*
- the name of the encoding scheme should be given as the attribute value, and should be in the form of an XML qualified name (QName):

```
<dc:identifier xsi:type="dcterms:URI">
  http://www.ukoln.ac.uk/
</dc:identifier>
```
The case of names

- *elements, element refinements* and *encoding schemes* should use the names specified in

DCMI Metadata Terms
http://dublincore.org/documents/dcmi-terms/

...note, the 15 DCMES element names all start with a lowercase letter...
Some examples

- Qualified DC
  
  example 4

- DC and IEEE LOM
  
  example 5

- DC, IMS and ODRL
  
  example 6

**HEALTH WARNING**

Examples 5 and 6 may seriously damage your interoperability!
Encoding DC in RDF
What is RDF?

- Resource Description Framework
- W3C recommendation for metadata
- model and syntax(es)
- RDF is commonly encoded as XML for use on the Web
- underpins the ‘semantic Web’

W3C - Resource Description Framework (RDF)
http://www.w3.org/RDF/
Why use RDF?

- RDF provides shared metadata ‘model’…
- …shared ‘meaning’
- metadata can be shared between applications that have little or no knowledge about each other
- e.g. an RDF-based bibliographic application can consume RDF-based geospatial metadata and have 'some' knowledge of what it means

...with (X)HTML and XML encodings, software applications must have 'understanding' hard-coded into them…
**DC in RDF**

- DC abstract model maps easily onto the RDF model (because RDF was the basis for it!)
- DC in RDF/XML syntax is an encoding of the RDF model in XML
- simple DC is similar to the non-RDF XML we've seen already…
- …but with the addition of `<rdf:RDF>` and `<rdf:Description>` container elements
- [example 7](#)
- qualified DC is too complex to cover here!
Practical examples – OAI and RSS
OAI-PMH

• OAI Protocol for Metadata Harvesting
• simple protocol for sharing metadata records between applications
• currently at version 2.0
• based on HTTP, XML, XML Schema and XML namespaces
• allows a harvester to ask a remote repository for some or all of its metadata records
OAI-PMH (2)

- simple DC is default (mandatory) record format
- supports any record format provided it can be encoded using XML (e.g. DC, IEEE LOM, MARC, ODRL, ...)

Open Archives Initiative
http://www.openarchives.org/
OAI-PMH example

• record from the American Memory repository at the Library of Congress

  http://memory.loc.gov/cgi-bin/oai2_0

• example 8

• ScreenCam of using the ‘repository explorer’

• GetRecord for record identifier

  oai:lcoa1.loc.gov:loc.gmd/g3701p.rr003570
**RSS**

- RDF Site Summary or Rich Site Summary (or even Really Simple Syndication)
- at least 3 different versions (0.91, 1.0 and 2.0)
- all based on XML but not compatible
- simple format for sharing news feeds on the Web
- RSS ‘channel’ – list of ‘items’
- channels updated by updating XML file
- RSS clients gather XML on regular basis
**RSS 1.0 and DC example**

- RSS 1.0 based on RDF
- most flexible and extensible of the RSS ‘family’ - not necessarily the most widely deployed
- can include DC in both ‘channel’ and ‘item’ descriptions
- example 9
- full documentation at:

  RDF Site Summary 1.0 Modules: Qualified Dublin Core
  http://web.resource.org/rss/1.0/modules/dctermss/
What have we learned?

- an abstract model for DC
- encoding DC in XHTML
- encoding DC in XML
- encoding DC in RDF/XML
- two practical examples
  - OAI Protocol for Metadata Harvesting
  - RSS
Questions?