



Basic Syntax

Tutorial 2 – DC-2006, Manzanillo, Mexico

Andy Powell, Eduserv Foundation
andy.powell@eduserv.org.uk
www.eduserv.org.uk/foundation





About me...

- Andy Powell
- Eduserv Foundation
 - UK charitable body with a mission to support the effective application of information and communication technology in education
- 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 HTML (15 mins)
- issues around DC in XML and DC in RDF (10 mins)
- 2 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/>

all under
revision
currently



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)
 - concepts (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* (usually a *value string*)



Value strings

- 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 can be inferred that "Ben is a **contributor** to the Book"
- sub-property** ←
property →



Encoding schemes

- *values* and *value strings* can be 'qualified' by using *encoding schemes*
- a *vocabulary encoding scheme* is used to indicate a set of which the *value* is a member
 - e.g. the *value* is a member of LCSH
- a *syntax encoding scheme* is used to indicate how the *value string* is structured
 - e.g. the *value string* is 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...

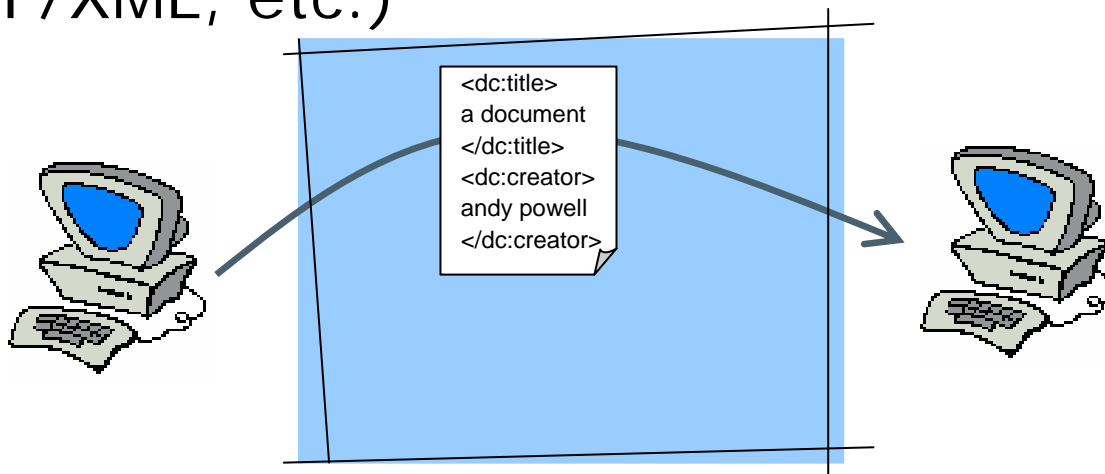


Description sets

- 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 (HTML meta tags, XML, RDF/XML, etc.)





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

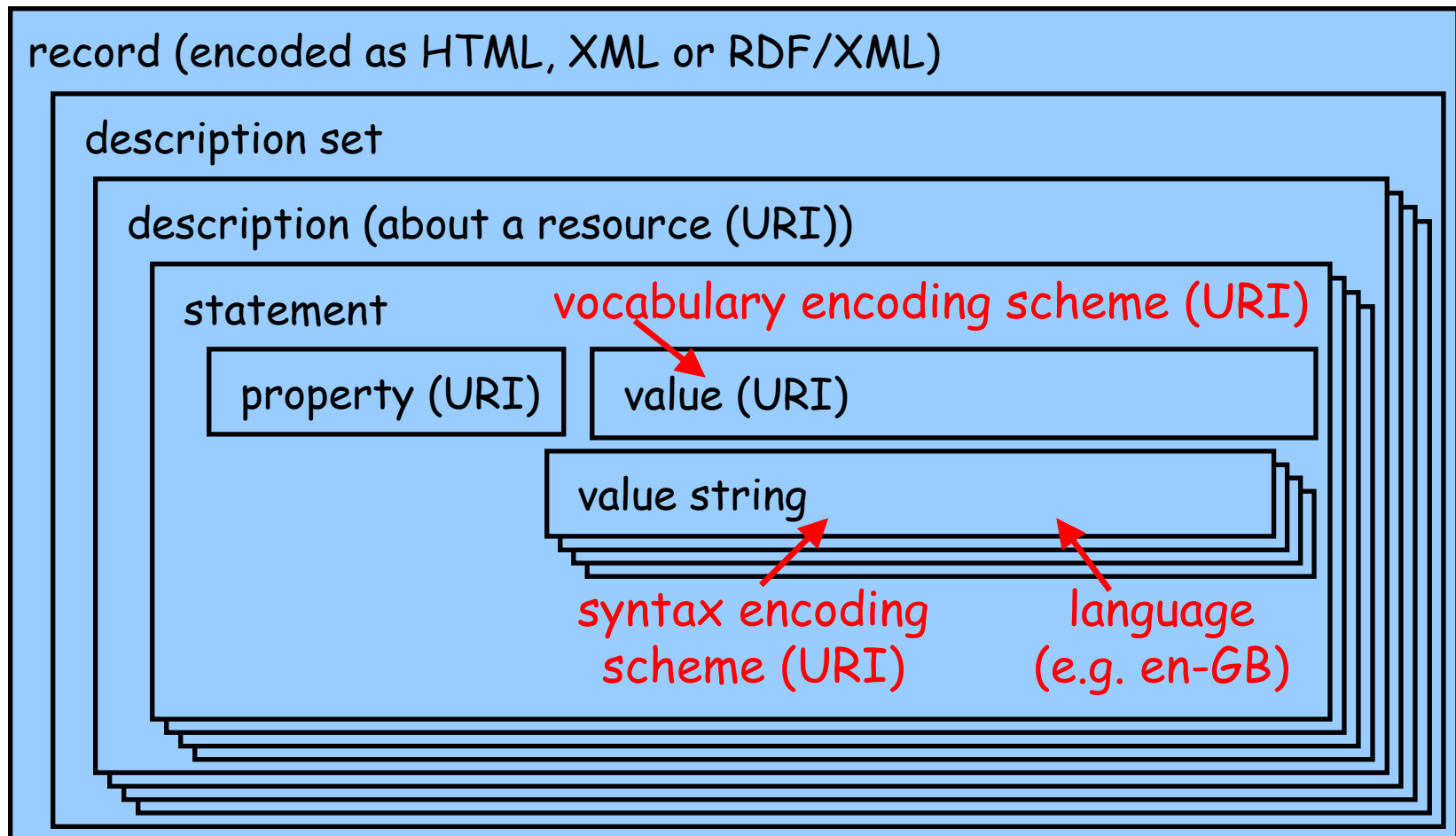
note that not everyone agrees with my definitions!

Dumb-down

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

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

Model summary





General summary

- Dublin Core is not just 15 metadata elements
- there are more than 50 properties approved by DCMI
- and the model allows you to declare and use your own properties as well
- descriptions sets are **very important**
- Dublin Core is not just about simple flat descriptions of single-entities
- the model allows you to group together descriptions of more complex sets of entities



DC in HTML



What is being described?

- a DC description embedded in an HTML document describes that document
- if you want to describe something else, don't embed it in the 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 >  
<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, `&`, `<`, `>`, 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:

```
<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:

```
<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  
devel opers" />
```

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



Element refinements

- use the same pattern for element refinements:

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

- for example:

```
<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:

```
<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:

```
<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/>



DC in XML and DC in RDF



What is XML?

- eXtensible Markup Language
- a generic markup language
- typically used to mark up text (XHTML), data (MathML, CML), images (SVG) and so on (for use on the Web)
- often used as the markup language for metadata
 - METS
 - MODS
 - EAD, ... and Dublin Core

DC in XML

- the current guidelines for how to encode DC in XML are being updated
- recent public comment period raised some issues that are currently being addressed
- subject for discussion at the DC Architecture meeting at DC-2006
- expect new “Expressing DC metadata using XML” DCMI Recommendation by Easter 2007



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
- the current guidelines for how to encode DC in XML are being updated
- subject for discussion at the DC Architecture meeting at DC-2006
- expect new “Expressing DC metadata using XML” DCMI Recommendation by Easter 2007



Practical examples OAI-PMH 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 3](#)
- [ScreenCam](#) of using the [‘repository explorer’](#)
- GetRecord for record identifier

`oai:loc1.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 4](#)
- full documentation at:

RDF Site Summary 1.0 Modules: Qualified Dublin Core

<http://web.resource.org/rss/1.0/modules/dcterms/>



What have we learned?

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



Questions and discussion