Providing Linked Data

Presented by:
Maribel Acosta
LINKED DATA LIFECYCLE
Linked Data Principles

1. Use URIs as names for things.
2. Use HTTP URIs so that users can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL).
4. Include links to other URIs, so that users can discover more things.
Core Tasks for Providing Linked Data

Based on the proposed LD lifecycles and the LD principles, we can identify 3 main tasks for providing LD:

① **Creating**: includes data extraction, creation of HTTP URIs, and vocabulary selection. *(LD principles 1 & 2)*

② **Interlinking**: involves the creation of (RDF) links to external data sets. *(LD principle 4)*

③ **Publishing**: consists of creating the metadata and making the data set accessible. *(LD principle 3)*
Agenda

1. Creating Linked Data
2. Interlinking Linked Data
3. Publishing Linked Data
4. Linked Data publishing checklist
CREATING LINKED DATA

Linked Data principles 1 & 2
Extracting the Data

• The data of interest may be stored in a wide range or formats:

- Spreadsheets or tabular data
- Databases
- Text

• Several tools support the process of mining data from different repositories, for example:

- Refine
- W3C R2RML
- Zemanta
- GATE
- DBpedia Spotlight
Naming Things: URIs

• All the *things* or distinct entities within the data must be named

• According to the Linked Data principles, the standard mechanism to name entities is the **URI**

• Designing **Cool URIs**:
  – Leave out information about the data regarding to: author, technologies, status, access mechanisms, ...
  – **Simplicity**: short, mnemonic URIs
  – **Stability**: maintain the URIs as long as possible
  – **Manageability**: issue the URIs in a way that you can manage

Source: http://www.w3.org/TR/cooluris/
Selecting Vocabularies

• Vocabularies model the **concepts** and the **relationship** between them in a knowledge domain.

• Terms from well-known vocabularies should be **reused** wherever possible.

• **New terms** should be define only if you can not find required terms in existing vocabularies.

• A large number of **vocabularies** in **RDF** are openly available, e.g., **Linked Open Vocabularies (LOV)**.
Selecting Vocabularies (2)

Linked Open Vocabularies

Source: http://lov.okfn.org/dataset/lov/

322 vocabularies (2013)
458 vocabularies (2014)
Selecting Vocabularies (3)

Linked Open Vocabularies: Analyzing MusicOntology

Vocabulary links:
Vocabularies referencing "mo" (7)

Vocabularies referenced by "mo" (25)

Source: http://lov.okfn.org/dataset/lov/details/vocabulary_mo.html
Other lists of well-known vocabularies are maintained by:

- **W3C SWEO Linking Open Data community project**
  
  http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/CommonVocabularies

- **Library Linked Data Incubator Group: Vocabularies in the library domain**
  
  http://www.w3.org/2005/Incubator/lld/XGR-lld-vocabdataset-20111025
Creating Linked Data (Quick Recipe)

Steps:

- **Source data** (database, text, ...)
- Names for things (instances): **URIs**
- **Vocabularies**
  - A new concept should be created only when it can not be found in existing vocabularies
INTERLINKING LINKED DATA

Linked Data principle 4
Interlinking Data Sets

• It’s one of the Linked Data principles!

4. Include links to other URIs, so that users can discover more things.

• Involves the creation of RDF links between two different RDF data sets:
  – Links at instance level (rdfs:seeAlso, owl:sameAs)
  – Links at schema level (RDFS subclass/subproperty, OWL equivalent class/property, SKOS mapping properties)

• Appropriate links are detected via link discovery
Interlinking Data Sets (2)

Challenges for link discovery

• Linked Data sets are **heterogeneous** in terms of vocabularies, formats and data representation

• Large range of knowledge **domains**

• **Scalability**: LD is composed of a large number of data sets and RDF triples, hence it is not possible to compare every possible entity pair

Source: Robert Isele. “LOD2 Webinar Series: Silk”
Interlinking Data Sets (3)

Challenges for link discovery

• An instance of link discovery is the entity resolution problem: *deciding whether two entities correspond to same object in the real world*

  • **Name ambiguities:** typos, misspellings, different languages, homonyms

  • **Structural ambiguities:** same concepts/entities with different structures. Requires the application of ontology and schema matching techniques
Interlinking Data Sets (4)

RDF data sets can be interlinked:

**Manually**
- Involves the manual exploration of LD data sets and their RDF resources to identify linking targets
- May not be feasible when the number of entities within the data set is very large

**(Semi-)Automatically**
- Using tools that perform link discovery based on linkage rules, for example: Silk, Limes and xCurator
owl:sameAs & rdfs:seeAlso (1)

• **owl:sameAs**
  - Creates links between individuals
  - States that two URIs refer to the same individuals

• **rdfs:seeAlso**
  - States that a resource may provide additional information about the subject resource

• Links in **MusicBrainz**:  
  – owl:sameAs is used for music artists  
  – rdfs:seeAlso is used for albums
Examples of owl:sameAs and rdfs:seeAlso

@prefix owl: <http://www.w3.org/2002/07/owl#>  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
@prefix dbpedia: <http://dbpedia.org/resource/>

<http://musicbrainz.org/artist/b10bbbfccf9e-42e0-be17-e2c3e1d2600d> owl:sameAs dbpedia:The_Beatles.

<http://musicbrainz.org/release/f8bf9e45-a1d1-42ab-8f3c-637b1762e5f7> rdfs:seeAlso dbpedia:All_you_need_is_love_(album).
SKOS

• Simple Knowledge Organization System
  – [http://www.w3.org/TR/skos-reference/](http://www.w3.org/TR/skos-reference/)

• Data model for knowledge organization systems (thesauri, classification scheme, taxonomies)

• SKOS data is expressed as RDF triples

• Allows the creation of RDF links between different data sets with the usage of **mapping properties**
SKOS: Mapping Properties

These properties are used to link SKOS concepts (particularly instances) in different schemas:

- **skos:closeMatch**: links two concepts that are sufficiently similar (sometimes can be used interchangeably)

- **skos:exactMatch**: indicates that the two concepts can be used interchangeably.
  - Axiom: It is a *transitive* property

- **skos:relatedMatch**: states an associative mapping link between two concepts
Example of SKOS exact match

@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix mo: <http://purl.org/ontology/mo/>
@prefix dbpedia-ont: <http://dbpedia.org/ontology/>
@prefix schema: <http://schema.org/>

mo:MusicArtist skos:exactMatch dbpedia-ont:MusicalArtist.


Example of SKOS close match

```xml
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix mo: <http://purl.org/ontology/mo/>
@prefix dbpedia-ont: <http://dbpedia.org/ontology/>
@prefix schema: <http://schema.org/>

mo:SignalGroup skos:closeMatch schema:MusicAlbum.

mo:SignalGroup skos:closeMatch dbpedia-ont:Album.
```
Integrity conditions

• Guarantee consistency and avoid contradictions in the relationships between SKOS concepts
Interlinking Linked Data (Quick Recipe)

Steps:

• Create **links** at instance level

• (Create **links** at schema level)

Interlinked RDF dataset
PUBLISHING LINKED DATA
Linked Data principle 3
Publishing Linked Data

Once the RDF data set has been created and interlinked, the publishing process involves the following tasks:

1. **Metadata** creation for describing the data set
2. Making the data set **accessible**
3. Exposing the data set in Linked Data **repositories**
4. **Validating** the data set
Describing RDF Data Sets

• Consists of providing (machine-readable) **metadata** of RDF data sets which can be processed by engines

• This information allows for:
  
  – Efficient and effective search of data sets
  
  – Selection of appropriate data sets (for consumption or interlinking)
  
  – Get general statistics of the data sets
Describing RDF Data Sets (2)

- The common language for describing RDF data sets is VoID (Vocabulary of Interlinked Data sets)

- Defines an RDF data set with the predicate `void:Dataset`

- Covers 4 types of metadata:
  - General metadata
  - Structural metadata
  - Descriptions of linksets
  - Access metadata
VoID: General Metadata

• General metadata is used by users to identify appropriate data sets.

• Specifies information about description of the data set, contact person/organization, the license of the data set, data subject and some technical features.

• VoID (re)uses predicates from the Dublin Core Metadata\(^1\) and FOAF\(^2\) vocabularies.


## General Information

Contains information about the creation of the data set

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcterms:title</td>
<td>Literal</td>
<td>Name of the data set.</td>
</tr>
<tr>
<td>dcterms:description</td>
<td>Literal</td>
<td>Description of the data set.</td>
</tr>
<tr>
<td>dcterms:source</td>
<td>RDF resource</td>
<td>Source from which the data set was derived.</td>
</tr>
<tr>
<td>dcterms:creator</td>
<td>RDF resource</td>
<td>Primarily responsible of creating the data set.</td>
</tr>
<tr>
<td>dcterms:date</td>
<td>xsd:date</td>
<td>Time associated with an event in the life-cycle of the resource.</td>
</tr>
<tr>
<td>dcterms:created</td>
<td>xsd:date</td>
<td>Date of creation of the data set.</td>
</tr>
<tr>
<td>dcterms:issued</td>
<td>xsd:date</td>
<td>Date of publication of the data set.</td>
</tr>
<tr>
<td>dcterms:modified</td>
<td>xsd:date</td>
<td>Date on which the data set was changed.</td>
</tr>
<tr>
<td>foaf:homepage</td>
<td>Literal</td>
<td>Name of the data set.</td>
</tr>
<tr>
<td>dcterms:publisher</td>
<td>RDF resource</td>
<td>Entity responsible for making the data set available.</td>
</tr>
<tr>
<td>dcterms:contributor</td>
<td>RDF resource</td>
<td>Entity responsible for making contributions to the data set.</td>
</tr>
</tbody>
</table>
Other Information

• **License of the data set:** specifies the usage conditions of the data. The license can be pointed with the property `dcterms:license`

• **Category of the data set:** to specify the topics or domains covered by the data set, the property `dcterms:subject` can be used

• **Technical features:** the property `void:feature` can be used to express technical properties of the data (e.g. RDF serialization formats)
VoID: Structural Metadata

• Provides high-level information about the internal structure of the data set

• This metadata is useful when exploring or querying the data set

• Includes information about resources, vocabularies used in the data set, statistics and examples of resources in the data set
Information about resources

- **Example resources**: allow users to get an impression of the kind of resources included in the data set. Examples can be shown with the property `void:exampleResource`

  ```turtle
  :MusicBrainz a void:Dataset;
  void:exampleResource <http://musicbrainz.org/artist/b10bbbfccf9e-42e0-be17-e2c3e1d2600d> .
  ```

- **Pattern for resource URIs**: the `void:uriSpace` property can be used to state that all the entity URIs in a data set start with a given string

  ```turtle
  :MusicBrainz a void:Dataset;
  ```
Vocabularies used in the data set

- The void:vocabulary property identifies the vocabulary or ontology that is used in a data set.

- Typically, only the most relevant vocabularies are listed.

- This property can only be used for entire vocabularies. It **cannot** be used to express that a subset of the vocabulary occurs in the data set.

```plaintext
:MusicBrainz a void:Dataset;
```
VoID: Structural Metadata

Statistics about a data set

Express numeric statistics about a data set:

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void:triples</td>
<td>Number</td>
<td>Total number of triples contained in the data set.</td>
</tr>
<tr>
<td>void:entities</td>
<td>Number</td>
<td>Total number of entities that are described in the data set. An entity must have a URI, and match the void:uriRegexPattern</td>
</tr>
<tr>
<td>void:classes</td>
<td>Number</td>
<td>Total number of distinct classes in the data set.</td>
</tr>
<tr>
<td>void:properties</td>
<td>Number</td>
<td>Total number of distinct properties in the data set.</td>
</tr>
<tr>
<td>void:distinctSubjects</td>
<td>Number</td>
<td>Total number of distinct subjects in the data set.</td>
</tr>
<tr>
<td>void:distinctObjects</td>
<td>Number</td>
<td>Total number of distinct objects in the data set.</td>
</tr>
<tr>
<td>void:documents</td>
<td>Number</td>
<td>Total number of documents, in case that the data set is published as a set of individual documents.</td>
</tr>
</tbody>
</table>

Source: http://www.w3.org/TR/void/#metadata
Partitioned data sets

- The `void:subset` property provides description of parts of a data set

  ```
  :MusicBrainz a void:Dataset;
  void:subset :MusicBrainzGroupMembers .
  ```

- Data sets can be partitioned based on classes or properties:
  - `void:classPartition` contains only instances of a particular class
  - `void:propertyPartition` contains only triples with a particular predicate

  ```
  :MusicBrainzGroupMembers a void:Dataset;
  void:classPartition [ void:class mo:MusicArtist .] ;
  ```
VoID: Describing Linksets

- **Linkset**: collection of RDF links between two RDF data sets

```owl
@PREFIX void:<http://rdfs.org/ns/void#>
@PREFIX owl:<http://www.w3.org/2002/07/owl#>

:DS1 a void:Dataset .
:DS2 a void:Dataset .
:LS1 a void:Linkset;
    void:linkPredicate owl:sameAs;
    void:target :DS1, :DS2 .
```

Example

```turtle
@PREFIX void:<http://rdfs.org/ns/void#>
@PREFIX owl:<http://www.w3.org/2002/07/owl#>

:MusicBrainz a void:Dataset .
:DBpedia a void:Dataset .

:MusicBrainz void:classPartition :MBArtists .
:MBArtists void:class mo:MusicArtist .

:MBArtists a void:Linkset;
   void:linkPredicate owl:sameAs;
   void:target :MusicBrainz, :DBpedia .
```
VoID: Access Metadata

The access metadata describes the methods of accessing the actual RDF data set

<table>
<thead>
<tr>
<th>Method</th>
<th>Predicate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI look up endpoint</td>
<td>void:uriLookupEndpoint</td>
<td>Specifies the URI of a service for accessing the data set (different from the SPARQL protocol)</td>
</tr>
<tr>
<td>Root resource</td>
<td>void:rootResource</td>
<td>URI of the top concepts (only for data sets structured as trees)</td>
</tr>
<tr>
<td>SPARQL endpoint</td>
<td>void:sparqlEndpoint</td>
<td>Provides access to the data set via the SPARQL protocol.*</td>
</tr>
<tr>
<td>RDF data dumps</td>
<td>void:dataDump</td>
<td>Specifies the location of the dump file. If the data set is split into multiple files, then several values of this property are provided.</td>
</tr>
</tbody>
</table>

* This assumes that the default graph of the SPARQL endpoint contains the data set. VoID cannot express that a data set is contained in a specific named graph. This can be specified with SPARQL 1.1. Service Description
Providing Access to the Data Set

The data set can be accessed via different mechanisms:

- Dereferencing HTTP URIs
- RDFa
- SPARQL endpoint
- RDF dump
Dereferencing HTTP URIs

- Allows for easily **exploring** certain resources contained in the data set

- **What to return for a URI?**
  - **Immediate description:** triples where the URI is the subject.
  - **Backlinks:** triples where the URI is the object.
  - **Related descriptions:** information of interest in typical usage scenarios.
  - **Metadata:** information as author and licensing information.
  - **Syntax:** RDF descriptions as RDF/XML and human-readable formats.

- Applications (e.g. LD browsers) render the retrieved information so it can be perceived by a user.

Source: *How to Publish Linked Data on The Web* - Chris Bizer, Richard Cyganiak, Tom Heath.
Example: Dereferencing [http://dbpedia.org/resource/The_Beatles](http://dbpedia.org/resource/The_Beatles)

The Beatles were an English rock band formed in Liverpool in 1960 and one of the most commercially successful and critically acclaimed acts in the history of popular music. The group's best-known lineup consisted of John Lennon (rhythm guitar, vocals), Paul McCartney (bass guitar, vocals), George Harrison (lead guitar, vocals) and Ringo Starr (drums, vocals).

The Beatles were a pop group that emerged in Liverpool, England, in the early 1960s, consisting of John Lennon, Paul McCartney, George Harrison, and Ringo Starr. They are considered one of the most influential and commercially successful bands in history. Their early music was characterized by its catchy melodies, harmonies, and Lennon-McCartney songwriting partnership. As they gained popularity, their sound evolved to include more experimental elements, such as The White Album (1968) and Abbey Road (1969), which featured complex arrangements and innovative production techniques. The Beatles' influence extended beyond music, impacting fashion, art, and culture. Their final album, Let It Be (1970), was released posthumously, following the band's disbandment in 1970.
RDFa

• RDFa = “RDF in attributes”

• Extension to HTML5 for embedding RDF within HTML pages:
  – The HTML is processed by the browser, the (human) consumer doesn’t see the RDF data
  – The RDF triples within the page are consumed by APIs to extract the (semi-)structured data

• It is considered as the bridge between the Web of Data and the Web of Documents

• It is a complete serialization of RDF
# RDFa: Attributes

<table>
<thead>
<tr>
<th>Attribute role</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>prefix</td>
<td>List of prefix-name IRIs pairs</td>
</tr>
<tr>
<td></td>
<td>vocab</td>
<td>IRI that specifies the vocabulary where the concept is defined</td>
</tr>
<tr>
<td>Subject</td>
<td>about</td>
<td>Specifies the subject of the relationship</td>
</tr>
<tr>
<td>Predicate</td>
<td>property</td>
<td>Express the relationship between the subject and the value</td>
</tr>
<tr>
<td></td>
<td>rel</td>
<td>Defines a relation between the subject and a URL</td>
</tr>
<tr>
<td></td>
<td>rev</td>
<td>Express reverse relationships between two resources</td>
</tr>
<tr>
<td>Resource</td>
<td>href</td>
<td>Specifies an object URI for the rel and rev attributes</td>
</tr>
<tr>
<td></td>
<td>resource</td>
<td>Same as href (used when href is not present)</td>
</tr>
<tr>
<td></td>
<td>src</td>
<td>Specifies the subject of a relationship</td>
</tr>
<tr>
<td>Literal</td>
<td>datatype</td>
<td>Express the datatype of the object of the property attribute</td>
</tr>
<tr>
<td></td>
<td>content</td>
<td>Supply machine-readable content for a literal</td>
</tr>
<tr>
<td></td>
<td>xml:lang, lang</td>
<td>Specifies the language of the literal</td>
</tr>
<tr>
<td>Macro</td>
<td>typeof</td>
<td>Indicate the RDF type(s) to associate with a subject</td>
</tr>
<tr>
<td></td>
<td>inlist</td>
<td>An object is added to the list of a predicate.</td>
</tr>
</tbody>
</table>
RDFa: Example

Extracting RDF from HTML

HTML (+RDFa):

```html
<div class="artistheader"
about="http://musicbrainz.org/artist/b10bbbc-cf9e-42e0-be17-e2c3e1d2600d#_
typeof="http://purl.org/ontology/mo/MusicGroup">

...  

</div>
```

RDF:

```xml
<http://musicbrainz.org/artist/b10bbbc-cf9e-42e0-be17-e2c3e1d2600d#_>
```
RDFa: Example

Extracting RDF from HTML

**HTML (+RDFa):**

```html
<div class="artistheader"
  about="http://musicbrainz.org/artist/b10bbbfacet9e-42e0-be17-e2c3e1d2600d#_
  typeof="http://purl.org/ontology/mo/MusicGroup">
...
</div>
```

**RDF:**

```xml
<http://musicbrainz.org/artist/b10bbbfacet9e-42e0-be17-e2c3e1d2600d#_>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
```
RDFa: Example

Extracting RDF from HTML

HTML (+RDFa):

```html
<div class="artistheader"
    about="http://musicbrainz.org/artist/b10bbbfccf9e-42e0-be17-e2c3e1d2600d#_"
    typeof="http://purl.org/ontology/mo/MusicGroup">
    ...
</div>
```

RDF:

```xml
<http://musicbrainz.org/artist/b10bbbfccf9e-42e0-be17-e2c3e1d2600d#_>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
```
Extracting RDF from MusicBrainz.org

http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d

The Beatles
~ Group

Annotation

The official name of “The White Album” is “The Beatles”; please do not add incorrectly titled releases to the database!

Please do not re-add individual mono remaster releases to their individual release groups. The mono remasters were never released separately, and are all already in the database as 13 mediums of the release the name “The Beatles in Mono”.

Annotation last modified on 2012-10-09 07:44 UTC.

Wikipedia

The Beatles were an English rock band formed in Liverpool in 1960. They became the most commercially successful and critically acclaimed act in the rock music era. The group’s best-known lineup consisted of John Lennon, Paul McCartney, George Harrison, and Ringo Starr. Rooted in skiffle and 1950s rock and roll, the Beatles later utilized several genres, ranging from pop belies to psychedelic rock, often incorporating classical and other elements in innovative ways. In the early 1960s, their enormous popularity first emerged as “Beatlemania”, but as their songwriting grew in sophistication, they came to be perceived by many fans and cultural observers as an embodiment of the ideals shared by the era’s sociocultural revolutions.

Continue reading at Wikipedia...

Discography

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Artist</th>
<th>Rating</th>
<th>Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>1963</td>
<td>Please Please Me</td>
<td>The Beatles</td>
<td>★★★★★</td>
<td>10</td>
</tr>
<tr>
<td>1963</td>
<td>With The Beatles</td>
<td>The Beatles</td>
<td>★★★★★</td>
<td>10</td>
</tr>
<tr>
<td>1964</td>
<td>Introducing... The Beatles</td>
<td>The Beatles</td>
<td>★★★★★</td>
<td>3</td>
</tr>
<tr>
<td>1964</td>
<td>Meet The Beatles!</td>
<td>The Beatles</td>
<td>★★★★★</td>
<td>1</td>
</tr>
</tbody>
</table>
RDFa: Example (2)

Extracting RDF from MusicBrainz.org

Source: http://www.w3.org/2007/08/pyRdfa/
RDFa: Example (2)

Extracting RDF from MusicBrainz.org

http://www.w3.org/2007/08/pyRdffa/extract?uri=http%3A%2F%2Fmusicbrainz.org%2Fartist%2Fb10bbfc-cf9e-42e0-be17-e2c3e1d2600d&format=nt

Watch the EUCLID screencast: http://vimeo.com/euclidproject

EUCLID - Providing Linked Data
RDF Dump

• An RDF dump refers to a file which contains (part of) a data set specified in an RDF serialization

• The data set can be split into several RDF dumps

• A list of available data sets available as RDF dumps can be found at:
  –  http://www.w3.org/wiki/DataSetRDFDumps
SPARQL Endpoint

• The **SPARQL endpoint** refers to the URI of the listener of the SPARQL protocol service, which handles requests for SPARQL protocol operations.

• The user submits **SPARQL queries** to the SPARQL endpoint in order to retrieve only a desired subset of the RDF data set.

• List of available SPARQL endpoints:
  • [http://www.w3.org/wiki/SparqlEndpoints](http://www.w3.org/wiki/SparqlEndpoints)
  • [http://sparqles.okfn.org/](http://sparqles.okfn.org/)
SPARQLES

EUCLID - Providing Linked Data

SPARQL ENDPOINTS STATUS > DATAHUB.IO

Last update: Tue Sep 02 2014 09:17:30 6MT+0000 (UTC)

Description:

http://sparqles.okfn.org/
Using Linked Data Catalogs

- Data catalogs, markets or repositories are platforms dedicated to **provide visibility** to a wide range of data sets from different domains

- Allow **data consumers** to easily find and use the data

- Usually the catalogs offer relevant **metadata** about the creation of the data set
Using Linked Data Catalogs (2)

How to publish an RDF data set into a catalog?

Create your own data catalog

- Recommended for big organizations/institutions aiming at providing a large number of data sets

Upload your data set into an existing catalog

- Allows data consumers to easily find new data sets

Use a data management system, for example:

- ckan

Common LD catalogs are:
- the Data Hub
- The Linking Open Data Cloud
Validating Data Sets

There are different ways to validate the published RDF data set:

Accessibility

- **Vapour** - Performs two types of tests: without content negotiation and requesting RDF/XML content
  
  http://validator.linkeddata.org/vapour

- **URI Debugger** - Retreives the HTTP responses of accessing a URI
  
  http://linkeddata.informatik.hu-berlin.de/uridbg/

- **RDF Triple-Checker** – Dereferences namespaces associated with the resources used in the document
  
  http://graphite.ecs.soton.ac.uk/checker/

Parsing & Syntax

- **W3C RDF/XML Validation Service** – Evaluates the syntax of RDF/XML documents and displays the RDF triples in it
  
  http://validator.linkeddata.org/vapour

- **W3C Markup Validation Service** – Checks syntactic correctness for web documents with RDFa markup
  
  http://validator.w3.org/

General validators

- **RDF:ALERTS** – Validates syntax, undefined resources, datatype and other types of errors
  
  http://swse.deri.org/RDFAlerts/
Validating Data Sets (2)

Example: Validating URIs with Vapour

Source: http://idi.fundacionctic.org/vapour
Validating Data Sets (3)

Example: Validating URIs with Vapour

Vapour Report

All tests passed!

Summary:

<table>
<thead>
<tr>
<th>Test requirement</th>
<th>Passed tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dereferencing resource URI (requesting RDF/XML)</td>
<td>3/3</td>
</tr>
<tr>
<td>Dereferencing resource URI (without content negotiation)</td>
<td>2/2</td>
</tr>
</tbody>
</table>

Source: http://idi.fundacionctic.org/vapour
Publishing Linked Data (Quick Recipe)

Steps:

• Create dataset **metadata**

• Provide **access mechanisms:**
  – RDF dump
  – SPARQL endpoint

• **(Expose** the dataset in a catalog)

• **Validate** everything!
PROVIDING LINKED DATA:
CHECKLIST
Providing Linked Data: Checklist (1)

Creating Linked Data

- All the relevant entities/concepts were effectively extracted from the raw data?
- Are all the created URIs dereferenceable?
- Are you reusing terms from widely accepted vocabularies?
Providing Linked Data: Checklist (2)

Interlinking Linked Data

- Is the data set linked to other RDF data sets?
- Are the created vocabulary terms linked to other vocabularies?
Providing Linked Data: Checklist (3)

Publishing Linked Data

- Do you provide data set metadata?
- Do you provide information about licensing?
- Do you provide additional access methods?
- Is the data set available in LD catalogs?
- Did the data set pass the validation tests?
Summary

In this chapter we studied:

• The Linked Data **lifecycle**:
  • 3 core tasks: creating, interlinking and publishing

• **Creation** of Linked Data:
  • Extracting relevant data, using URIs to name entities and selecting vocabularies and expressing the data using the RDF data model

• **Interlinking** Linked Data:
  • Challenges of link discovery, using Silk to create links between two data sets and using SKOS links

• **Publishing** Linked Data:
  • Creation of data set metadata; publishing the data set via RDF dumps, SPARQL endpoints or RDFa; uploading the data set to a LD catalog
For exercises, quiz and further material visit our website: http://www.euclid-project.eu

Ebook
- HTML
- Kindle
- iBooks
- ePUB

Course
- HTML
- iTunes U

Other channels:
- @euclid_project
- euclidproject
- euclidproject