DELIVERABLE

Project Acronym: DCA

Grant Agreement number: 270927

Project Title: Digitising Contemporary Art

D5.3 Enrichment module and POC

Revision: V1.1

Author(s): Sam Coppens (IBBT/MMLab); Erik Mannens (IBBT/MMLab)
Reviewer(s): Rony Vissers (PACKED vzw); Joris Janssens (PACKED vzw)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisation	Description
V0.1	31/08/2012	SC, EM	IBBT	First draft
V0.2	03/09/2012	RV, JJ	PACKED vzw	First internal review
V0.3	13/09/2012	SC, EM	IBBT	Amended draft
V1.0	13/09/2012	RV, JJ	PACKED vzw	Final review
V1.1	17/01/2013	RV	PACKED vzw	Proofreading by native English speaker

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Content Table

Executive Summary

In this deliverable, we discuss how to publish data as Linked Open Data in practice. This means we show how to do it, which steps to take and which open source tools can be used for the different steps.

The basic steps in publishing data as Linked Open Data are:

- select appropriate RDF model to publish the data;
- choose a Linked Open Data server infrastructure;
- transform the data to RDF;
- enrich the data.

Each of the steps is discussed in detail and demonstrated by our proof of concept. An important step in publishing Linked Open Data is to enrich the data. In this deliverable, special attention will be given to the enrichment module.

Finally, we'll present this deliverable with a workflow recommendation for the DCA partners to publish their data as Linked Open Data.

1. Linked Open Data publication

Introduction

The main goal of Linked Open Data (LOD) (1) is to allow people to share structured data on the Web as easily as they share documents today. It actually refers to a style of publishing and interlinking structured data on the Web. The main recipe for LOD is RDF (2), the Resource Description Framework. The structured data is published as RDF data (using an RDF data model) and RDF links are used to link data from different data sources. This way, the LOD on the Web creates a giant global graph, across which all the published data is connected to each other. It's also called the Web of Data. Clients can easily discover and consume data through it.

LOD stipulates four basic principles [1]:

- The first principle is that we have to identify the items of interest in our domain first of all. Such items are the resources, which will be described in the data.
- The second principle is that those resources have to be identified by HTTP URIs and one should avoid schemes such as Uniform Resource Names (URNs) and Digital Object Identifiers (DOIs).
- The third principle is to provide useful information when accessing an HTTP URI.
- The fourth rule is to provide links to the outside world, i.e. to connect the data with that from other datasets in the Web of Data. This makes it possible to browse data from a certain server and receive information from another server. In other words, by linking the data with that of other datasets, the web becomes one huge database, called the Web of Data.

To demonstrate LOD publication, a proof of concept is set up. This is provided by an LOD server which publishes and enriches the DCA content. The URL of the LOD server is http://dca.test.ibbt.be:8080/. At present, the LOD server puts some example records online as EDM records. In the future, the DCA content will be harvested, enriched and published as LOD, as soon as *Europeana* has harvested all the content.

Preliminaries / requirements

When publishing structured data as LOD, first of all we need to select the items, which will be published as LOD. These resources will be identified using HTTP URIs, so that the resources become available on the Web using standard HTTP mechanisms (e.g., using a web browser). These two conditions comply with the first two principles for publishing LOD.

The selected and identifiable resources need of course to be described. For this, as explained earlier, we use RDF models. This is a first preliminary for publishing LOD: a metadata model for publishing the selected resources as RDF to the Web. This refers to the third principle of publishing LOD.

Every described resource can have multiple representations, e.g., HTML (3), RDF/XML (4), Turtle (5), etc. These are there to provide useful information on the resource. Assuming that the client is a human using a web browser to look up some information about a resource, the server publishing the LOD will then serve an HTML page, because HTML is designed to present information to a human. When a machine agent wants to look up information about the resource, the server will provide an RDF/XML representation about the resource, because RDF/XML is machine-readable. This content

[1] http://www.w3.org/DesignIssues/LinkedData.html

negotiation is the responsibility of the LOD server that will serve the right representation about a resource, based on the preferences of the client. This also refers to the third principle of LOD.

In order to provide links to external data sources, the fourth principle of publishing LOD, the published content needs to be enriched, via an enrichment module. This is a fourth preliminary for publishing LOD, which will interconnect all the published LOD content in order to create the Web of Data.

To summarise, the following entities are preliminary for LOD publication:

- a metadata model;
- an LOD server;
- an enrichment module.

These three entities will be discussed in detail in the next sections.

2. Linked Open Data Model

Purpose

When publishing data on the Web, you need a data model to describe your data. The same holds true for publishing LOD, which relies on RDF models to describe the data. In RDF models, everything is described using triples, as explained in more detail in deliverable 3.1 *Metadata implementation guidelines for digitised contemporary artworks* and deliverable 3.2 *Recommendation on contextualisation and interlinking digitised contemporary artworks*. These models are targeted at making links. One can easily interconnect pieces of information, each using their own RDFS (6) / OWL (7) ontology. One can even mix the ontologies to describe your pieces of information.

When developing a data model, one should choose those ontologies that best fit one's needs or one can make one's own schema, but then one needs to link it to popular ontologies to make one's data more interoperable. There exist various ontologies, each targeted at describing a specific thing or piece of information. In the next section, we provide examples of ontologies that are well known and much used, classified by their object of description.

RDF/RDFS/OWL ontologies

Cross-domain

Dublin Core (DC, (8)): a very generic model that can describe basic features (who, what, where, and when) about anything. The core model exists of fifteen properties.

Dolce Ultra Lite (DUL, (9)): a lightweight upper ontology that describes general concepts across all knowledge domains. Such upper ontology is often used to unify data, described using various ontologies.

Persons/organisations

Friend-Of-A-Friend (FOAF, (10)): a vocabulary for describing persons and organisations, e.g., the first and last name of a person, their social network accounts, contact information, etc.

Locations

Basic Geo (WGS84 lat/long) Vocabulary (11): a very basic vocabulary to express the longitude and latitude of a location. It is very simple but widely spread and powerful in combination with Dublin Core or FOAF.

Events

Linking Open Description of Events (LODE, (12)): a basic model that can describe events. It describes the event itself, when it took or will take place, where, which actors are involved, etc.

Cultural heritage

Europeana Data Model (EDM, (13)): a model that is targeted at describing cultural heritage information as LOD. It is developed by and for *Europeana* and is closely related to the LIDO schema.

OpenART (14): an event-driven ontology produced to describe 'art world' datasets. The ontology is split into a number of parts to allow greater re-usability. It's linked to the DUL ontology for greater applicability and interoperability.

Products

GoodRelations (15): a model that describes products and services offered for e-commerce. It is able to describe all the details of the products, such as price, stock, etc.

Proof of concept, developed within the framework of DCA

For the proof of concept, the EDM model is employed to publish DCA data on contemporary artworks. The EDM model was chosen, because it was especially designed by *Europeana* to publish their content on cultural heritage as LOD.

To ingest the data, the proof of concept relies on OAI-PMH (16) for harvesting. OAI-PMH is a harvesting protocol, which is supported by many aggregators and platforms of cultural heritage institutions. The OAI-PMH protocol was discussed in detail in deliverable 5.1 *Assessment of the different aggregation platforms and their aggregation requirements*.

Many DCA content partners will provide their data to *Europeana* using the MINT tool[2] for mapping and harvesting. From this MINT platform, we can ingest the data. The formats we accept using OAI-PMH are EDM and LIDO (17). When LIDO records are received, they are mapped to EDM records using an XSLT (18) mapping schema. For this XSLT, we followed EDM mapping guidelines, published by *Europeana* Professional[3]. Only the metadata are ingested into the proof of concept, no Web resources (digital representations).

When mapping to an RDF model that needs to be published as LOD, attention needs to be given to the used URIs. These all need to be HTTP URIs. Every instance of an RDF class needs to have an HTTP URI. Once an URI has been established the content negotiation can take place, when the data is published as LOD. This content negotiation is discussed in detail in the next section. For the form of URIs we have chosen for the following composition:

[BASE URI LOD Server (http://dca.test.ibbt.be:8080)]/resource/[class name]/ID

In EDM the following class instances are present:

Core EDM classes

• edm:ProvidedCHO	The provided cultural heritage object.
• edm:WebResource	The web resource that is a digital representation.
• edm:Aggregation	The aggregation that groups classes together.

Contextual EDM classes

• edm:Agent	The related agents (persons, organisations).
• edm:Place	The related locations.
• edm:Timespan	The related timespans.
• skos:Concept	The related SKOS (19) concepts.

Apart from the SKOS instances, which are hosted elsewhere, all the instances of these classes need HTTP URIs in the form shown above.

[2] http://mint.image.ece.ntua.gr/redmine/projects/mint/wiki
[3] http://pro.europeana.eu/documents/900548/ea68f42d-32f6-4900-91e9-ef18006d652e

3. LOD server

Purpose

Once we have the RDF model and instances of it, we need to store this metadata and need to publish it as LOD. The LOD server is responsible for publishing the information. This means it is responsible for giving access to the published data (via standardised HTTP methods) and for querying it. Both functions of the LOD server are discussed in detail in the sections below.

Access

As explained earlier, the LOD server needs to provide useful information to the client. This information on the published resources is represented using RDF. A normal web browser should not serve raw RDF to an HTML browser. Therefore, it will serve, alongside the RDF representation of the information, an HTML one. As such, the LOD server should serve representations of the data that can be understood by both humans (HTML) and machine agents (RDF). To achieve this, there are two main approaches:

- Embedding RDFa (20) in the HTML: In this approach, the server serves only one representation of the information, i.e., the HTML representation. The RDF data is embedded within it, using RDFa. RDFa is a microformat for capturing RDF information.
- Content Negotiation: Here, the LOD server will serve two representations of the published data: the HTML representation and the RDF one. The HTTP client sends an HTTP header in each request in which the client indicates which representation he prefers. Based on this preference, the content negotiation can take place. In practice, this means that every resource described by an RDF scheme has to be identified by an HTTP URI, (e.g., http://dbpedia.org/resource/Gerhard_Richter). Every resource should also have two representations: an XHTML (human-readable) and an RDF representation (machine-readable). Every representation also has to be identified by an HTTP URI (e.g. http://dbpedia.org/page/Gerhard_Richter) for the XHTML representation and for RDF representation (e.g. http://dbpedia.org/data/Gerhard_Richter). When coming across the HTTP URI of a resource, the LOD server determines which representation should be served, based on information in the accept header of the user's client, and then redirects the client to the appropriate representation using 303 redirect and content negotiation. This is shown in figure 1.

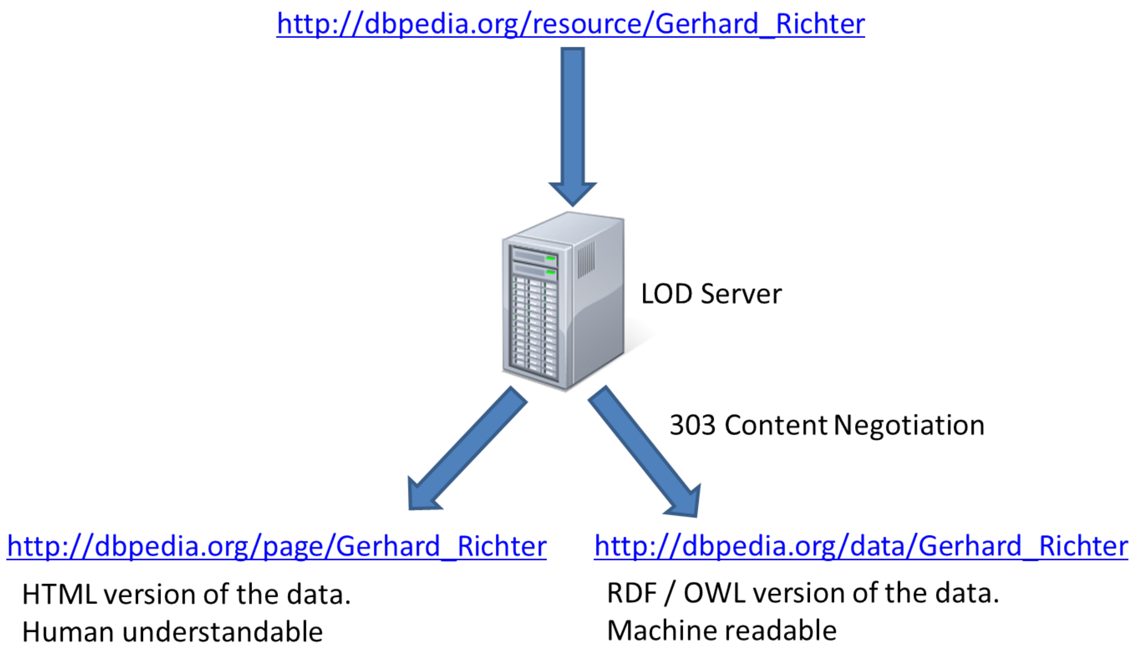

http://dbpedia.org/resource/Gerhard_Richter
LOD Server
303 Content Negotiation
http://dbpedia.org/page/Gerhard_Richter
HTML version of the data.
Human understandable
http://dbpedia.org/data/Gerhard_Richter
RDF / OWL version of the data.
Machine readable

Figure 1: content negotiation LOD server

Query

Another task for the LOD server is to make sure the content it publishes can be queried. SPARQL (21) was designed specifically to this end. SPARQL is a query language and data access protocol for the Semantic Web. It is defined in terms of the W3C's RDF data model and will work for any data source that can be mapped into RDF. As such, the LOD server is also responsible for publishing a SPARQL endpoint, at which SPARQL queries can be fired and results are returned.

Providing a SPARQL endpoint as LOD server is very important. It opens up your data and lets other data sources use it to enrich their own. Of all query languages, those that emulate SQL syntactically have probably been the most popular and widely implemented. This is perhaps surprising given the very different models that lurk behind relational databases and RDF. Familiarity with syntax has no doubt contributed to such success. SPARQL follows this well-trodden path, offering a simple, reasonably familiar (to SQL users) SELECT query form.

The SPARQL query below returns the names and emails of every person whose information is published by a LOD server. The persons are described using FOAF ontology. Notice the similarity with SQL, except for the WHERE clauses, which are formulated using RDF.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE
{
	?person a foaf:Person.
	?person foaf:name ?name.
	?person foaf:mbox ?email.
}
```

Listing 1: example SPARQL query

Open source tools

There exist a lot of open source frameworks which provide an LOD server. These are targeted at publishing the data and to providing a SPARQL endpoint for the data. In this section, we provide a list of open source tools that can be used for putting up an LOD server.

The tools are categorised according to purpose. We make a distinction between triple stores, Linked Data frontends, SPARQL endpoints and relational database LOD publishers. There are two practical ways of publishing RDF data as LOD:

- One can take an out-of-the-box solution: These platforms provide a triple store for storing your RDF data, a SPARQL endpoint to query the RDF data over HTTP, and a Linked Data frontend to publish RDF data using content negotiation. An example of such a platform is Openlink Virtuoso.
- One can compile one's own LOD server: For this one must first choose a triple store for storing the RDF data. Most of the triple stores just provide storage for your RDF data. They can all handle SPARQL queries, but this doesn't mean they make an SPARQL endpoint available over HTTP. Once one has a triple store one needs a server to set up a SPARQL endpoint over HTTP and a Linked Data frontend to publish data as LOD.

Triple Stores

Triple stores are RDF databases. They store the RDF data and provide a SPARQL endpoint for the data. Most do not offer public access to the data. This means they do not publish the data; they only store it and make sure it can be queried using SPARQL. This means that you have to transform your data to RDF yourself and provide a Linked Data frontend yourself, which will do the content negotiation.

Jena[4]

Jena is a Java RDF API and toolkit. It is a Java framework to construct semantic web applications. It allows you to store data in-memory, and to query the data. It also allows one to reason over the data using RDFS or OWL. Jena is no server that publishes your data or provides a SPARQL endpoint.

Sesame[5]

Sesame is very similar to Jena. It is a Java RDF database, with support for RDFS reasoning and SPARQL querying. It has a wide range of tools for developers.

TDB[6]

TDB is part of Jena and provides persistent RDF storage and query. TDB can be used as a high performance RDF store on a single machine. TDB is not a server, nor does it provide a SPARQL endpoint.

Openlink Virtuoso (open source edition)[7]

Openlink Virtuoso is a general-purpose database. It is actually relational and also provides a Linked Data interface and a SPARQL endpoint for the data. It is also a server, publishing data as LOD and providing a SPARQL endpoint.

OWLIM[8]

[4] http://jena.apache.org/
[5] http://www.openrdf.org/
[6] http://jena.apache.org/documentation/tdb/index.html
[7] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/

OWLIM is a very scalable triple store. It also provides an inference engine and SPARQL query engine.

Linked Data frontends

Linked Data frontends usually take their data from an SPARQL endpoint and provide a Linked Data interface to the data from it. These frontends take care of the content negotiation needed. Some triple stores are also servers and already provide a Linked Data interface, e.g., Openlink Virtuoso, while others don't, e.g., TDB or Jena.

Pubby[9]

Pubby can be used to provide a Linked Data interface to SPARQL endpoints. When you already have your data in a triple store, Pubby provides the Linked Data interface (with content negotiation).

Linked Data Pages[10]

Linked Data Pages is a Linked Data publishing framework in PHP. It relies, just like Pubby, on a SPARQL endpoint for publishing the data.

SPARQL endpoints

Triple stores that are also servers. They mostly provide a SPARQL endpoint already. Some triple stores, e.g., Jena or Sesame, do not provide this functionality. These tools will implement a SPARQL endpoint. They mostly have a binding to the data in the form of a model, which is implemented by Jena.

SPARQLer[11]

SPARQLer is a general-purpose SPARQL processor and query validator, based on Jena. It can be used to provide a local SPARQL endpoint.

Joseki[12]

Joseki is an HTTP engine that supports SPARQL. It can be used to put up SPARQL endpoints.

Fuseki[13]

Fuseki is a SPARQL server. It replaces Joseki and provides REST-style SPARQL support. It can be easily used in combination with Jena or TDB.

Su4j[14]

Su4j is a Jena-based servlet implementation of SPARQL. It can be used to provide a SPARQL endpoint.

[8] http://www.ontotext.com/owlim
[9] http://www4.wiwiss.fu-berlin.de/pubby/
[10] https://github.com/csarven/linked-data-pages
[11] http://sparql.org/
[12] http://www.joseki.org/
[13] http://jena.apache.org/documentation/serving_data/index.html
[14] https://bitbucket.org/fundacionctic/su4j/wiki/Home

Relational database LOD publishers

When your data comes from a relational database, there exist solutions that put an extra layer on top of the database to provide a Linked Data interface and SPARQL endpoint. This way, the data from your relational database doesn't need to be transformed and migrated to a triple store. A disadvantage of this method is that such a triple store is not that flexible in storing data using various ontologies, because they have to be reflected in the database structure.

*D2R Server*15

D2R Server is a tool for publishing the data from relational databases on the semantic Web. The tool provides a SPARQL endpoint for the data and a Linked Data interface.

Openlink Virtuoso

Openlink Virtuoso is actually a relational database with an extra layer for publishing and querying linked data from relational databases, as explained above.

Proof of concept, developed within the DCA framework

For the proof of concept within the DCA framework, we made our own LOD server, based on tools described above. The LOD server consists of the following components:

- an application server;
- a triple store;
- a SPARQL endpoint;
- a Linked Data frontend.

The figure below gives an overview of our architecture for publishing the LOD. Each of the following components is discussed in detail in the following section.

[15] http://d2rq.org/d2r-server

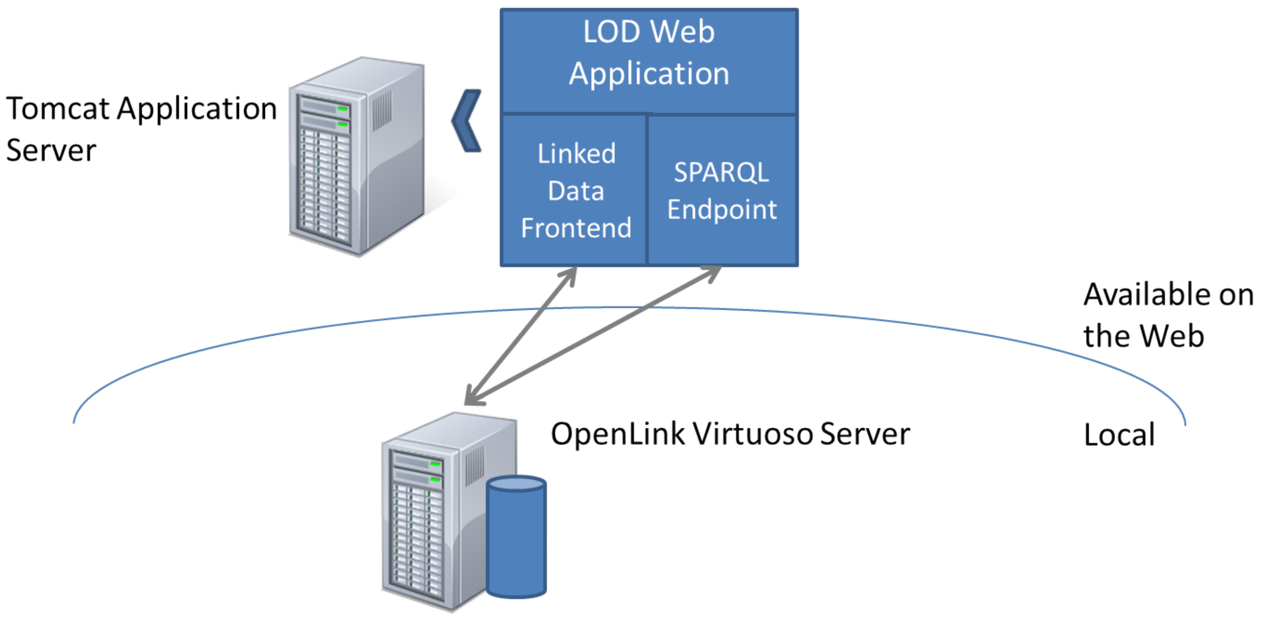


Figure 2: LOD server architecture of the DCA POC

Application server

We have chosen Apache Tomcat[16] as application server. It is an open source application server, implementing Java Servlet and JavaServer Pages technologies. It is easy to use and has a huge user community.

Triple store

Our triple store choice is Openlink Virtuoso. Virtuoso Universal Server is a middleware and database engine hybrid that combines the functionality of traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server functionality in a single system. Virtuoso is as such a universal server. The open source edition of Virtuoso Universal Server is also known as OpenLink Virtuoso. Based on the BSBM benchmark, Openlink Virtuoso offers a good scalability and query performance. It is just used for storing the RDF data.

SPARQL endpoint

Openlink Virtuoso is also a server and offers a SPARQL endpoint over HTTP directly. However, we've developed our own SPARQL endpoint. This way, the Virtuoso server could stay local and only the web application publishing LOD is opened up for the Web. At the same time, we can maintain more control over the resources that can be queries using the SPARQL endpoint.

Joseki was used to implement the SPARQL endpoint. Joseki just needs to make a connection to the triple store in order to retrieve the RDF data model and then publish a SPARQL endpoint for it. It is a means of keeping control of what is in the model for the SPARQL endpoint and what is not.

[16] http://tomcat.apache.org/

Linked Data frontend

In the proof of concept, we made our own Linked Data frontend. To create our own Linked Data interface, we implemented three servlets. One was for content negotiation, one for providing HTML representation of the data, and the third one was to deliver the RDF model in the asked representation (RDF/XML, n-tuples or json). The Figure below gives a schematic overview of the servlets and a piece of example code is shown for content negotiation.

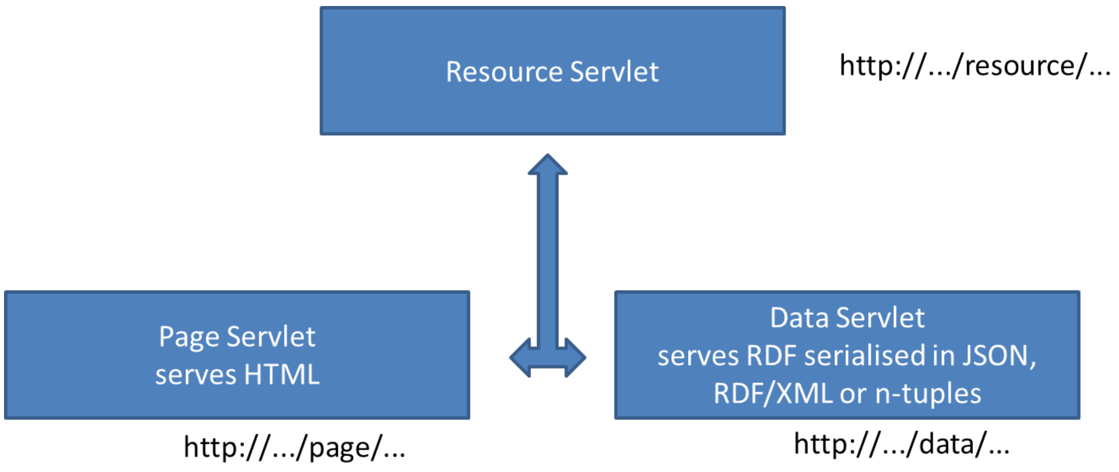


Figure 3: content negotiation and URI templates DCA PoC

Example code for the Resource servlet to do the content negotiation:

```
import java.io.IOException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import de.fuberlin.wiwiss.pubby.negotiation.ContentTypeNegotiator;
import de.fuberlin.wiwiss.pubby.negotiation.MediaRangeSpec;
import de.fuberlin.wiwiss.pubby.negotiation.PubbyNegotiator;

public class ResourceServlet extends HttpServlet {

	public void doGet(HttpServletRequest request, HttpServletResponse response)
			throws IOException {
		String relativeResourceURI = request.getRequestURI().substring(
				request.getContextPath().length()
						+ request.getServletPath().length());
		// Some servlet containers keep the leading slash, some don't
		if (!"".equals(relativeResourceURI)
				&& "/".equals(relativeResourceURI.substring(0, 1))) {
			relativeResourceURI = relativeResourceURI.substring(1);
		}
		if (request.getQueryString() != null) {
			relativeResourceURI = relativeResourceURI + "?"
					+ request.getQueryString();
		}

		response.addHeader("Vary", "Accept, User-Agent");
		ContentTypeNegotiator negotiator =
PubbyNegotiator.getPubbyNegotiator();
```

```
            MediaRangeSpec bestMatch = negotiator.getBestMatch(request
                        .getHeader("Accept"), request.getHeader("User-Agent"));
            if (bestMatch == null) {
                  response.setStatus(406);
                  response.setContentType("text/plain");
                  response.getOutputStream().println(
                              "406 Not Acceptable: The requested data format is
not supported. "
                                          + "Only HTML and RDF are available.");
                  return;
            }

            response.setStatus(303);
            response.setContentType("text/plain");
            String location;

            if ("text/html".equals(bestMatch.getMediaType())) {
                  location = pageURL(relativeResourceURI); // method to generate
page URL from relativeResourceURI
            } else {
                  location = dataURL(relativeResourceURI); // method to generate
data URL from relativeResourceURI

            }
            response.addHeader("Location", location);
            response.getOutputStream().println(
                        "303 See Other: For a description of this record, see "
                                    + location);
      }
}
```
Listing 2: example Java code of content negotiation

4. LOD enrichment

Purpose

Enriching the data will actually interlink it with data from other data sources. These enrichments will therefore link your data graph to the giant, global data graph. Data can easily be discovered and consumed through it.

The figure below shows how the giant global data graph[17] looks today. Every dot in the graph is a dataset being published as LOD. The interconnections to the LOD datasets are enrichments. The content can be categorised. There is data on media (blue), geographic data (orange), data on publications (green), user generated data (red), government data (turquoise), cross-domain data (light blue) and life sciences data (purple).

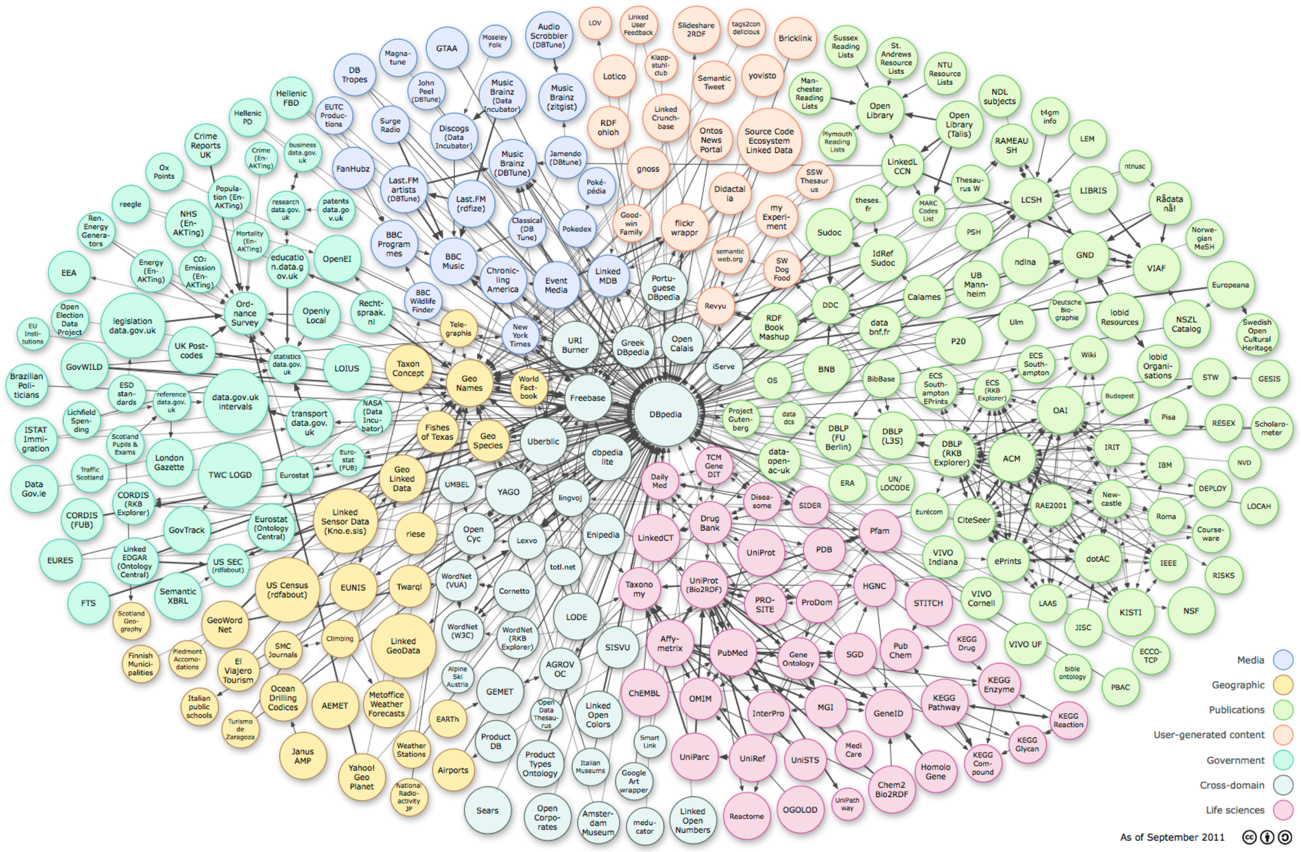


Figure 4: giant global data graph

The enrichments serve a dual purpose. Firstly, they help to disambiguate the data. If you have a resource describing a person called Steve McQueen, for example then this information is not enough to disambiguate Steve McQueen from Steve McQueen (the American actor), or Steve McQueen (the British visual artist / filmmaker). When the resource is linked to an external one, then this information helps to disambiguate it. When the resource is linked to http://viaf.org/viaf/9855712, we know we are talking about Steve McQueen, the American actor. When the resource is linked to http://viaf.org/viaf/96536223, the resource denotes Steve McQueen, the British visual artist / filmmaker.

[17] http://linkeddata.org/

A second purpose for interlinking data is to bring in some extra information about the resource. Let's look at the example of Steve McQueen, the British visual artist / filmmaker. If we only have a name and the year of birth for Steve McQueen in the triple store, then linking the resource to http://viaf.org/viaf/96536223 brings in extra information on him, such as titles of publications and unique identifiers from the Getty's Union List of Artist Names (www.getty.edu/research/tools/vocabularies/ulan/). The more your data is interlinked with external resources, the more value the data obtains and the more meaningful yours becomes. By enriching your data, you add more context information to it.

A key ingredient for discovering enrichments is SPARQL. Via a SPARQL endpoint the published datasets can be queried to detect enrichments. When publishing your data as LOD, a SPARQL endpoint is therefore crucial. It lets other data sources link to your dataset.

There exist various approaches for enriching data. You can do it manually, but this only holds for very small datasets. Bigger datasets ask for more automatic approaches. You can develop your own enrichment module, with your own specific enrichment algorithms, or you can use an enrichment tool, which will discover enrichments for you.

When developing your own enrichment module, the enrichment algorithm can be pattern-based or property-based. In the pattern-based approach, the enrichments are based on generally accepted naming schemas. A typical example of one such naming schema is the ISBN numbers. They uniquely identify publications/books. One can use them to search for remote records via SPARQL, which describe the same publication (publications with the same ISBN number). You can then link your records to the remote records discovered using the owl:sameAs property.

Another approach in developing your own enrichment algorithm is the property-based approach. With it, you can use certain properties of an entity to discover the same entity in a remote dataset. For instance, a person can't be uniquely identified by just his/her name. But the combination of the name, a date and place of birth will uniquely identify that person. You can use SPARQL to query remote datasets and look for persons with the same name, date and place of birth. Instead of relying on a unique naming convention to identify things, as in the pattern-based approach, you rely on a unique combination of properties to identify the same thing.

Open Source Tools

SILK[18]

SILK is a framework for discovering relationships between data items within different data sources. It is based on SPARQL. Via a configuration file, you can define your own SPARQL queries to discover relationships. These can be pattern-based or property-based.

Proof of concept, developed within the framework of DCA

For enriching the content of our PoC, we have developed our own property-based enrichment module. Our enrichment algorithm will make a distinction between the different types of entities, e.g., persons or locations, present in a record.

In D3.1 *Metadata implementation guidelines for digitised contemporary artworks* we gave some metadata best practices for content dissemination through *Europeana*. All created records from the

[18] http://www4.wiwiss.fu-berlin.de/bizer/silk/

contemporary art institutions within the DCA project are harvested and mapped to LIDO. *Europeana* can then harvest hese LIDO records. A list of properties for every sort of item that should be provided by the content partner was published in the deliverable. The tables below show the property lists for every sort of item, i.e., artworks, digital resources (surrogates), events, and actors. The values for fields denoted with an asterisk should be taken from a controlled vocabulary.

Artwork	Minimum	Recommended	Additional
ID	X		
Title	X		
Date		X	
Type	X		
Description		X	
Place		X	
Measurements			X
Collection		X	
Rights		X	
Language		X	
Subjects/keywords*	X		
Events			X
Surrogates			X

Surrogate	Minimum	Recommended	Additional
URL	X		
Description		X	
Language			X
Date		X	
Type*		X	
Rights		X	
Measurements			X
Format (mimetype)		X	

Event	Minimum	Recommended	Additional
Title	X		
Date		X	
Type*	X		
Description		X	
Place			X
Actor			X
Language		X	

Actor	Minimum	Recommended	Additional
Name	X		
Role*		X	
Place			X
Biography			X

Year of Birth		X	
Year of Death			X

Listing 2: table of recommended fields to be offered by the contemporary art institutions

These are also the properties used for interlinking data. This means that the properties will be used for building SPARQL queries. For every type of entity we want to enrich, we specify a property-based SPARQL query to look for enrichments. The entities we will be enriching are:

- locations;
- persons;
- artwork type;
- artwork.

For every type, we specify a set of properties to identify the entity uniquely. These properties can then be used to build queries for remote datasets. The datasets that are candidate for enrichment are discussed in the next Section.

Locations

Identifying properties:

- name;
- geographical coordinates.

Unfortunately, the geographical coordinates were not present in the content partners' databases. We can therefore only use the name of the location. Sometimes the country is also present in the content partners' databases. This property is used as extra context information.

```
PREFIX dbpediaprop: <http://live.dbpedia.org/property/dateOfBirth>
PREFIX dbpedia: <http://dbpedia.org/resource/ >
PREFIX dbpediaowl: <http://live.dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?place
WHERE
{
?place a dbpediaowl:Place.
?place rdfs:label 'Rijeka'@en.
?place dbpediaowl:country dbpedia:Croatia.
}

Result: <http://dbpedia.org/resource/Rijeka>
```

Listing 3: example SPARQL query for DBpedia[19] looking for a place Rijeka

If the country is not specified, DBpedia could return more than one result. To solve this, we need human intervention to pick out the right one. This shows why it is important to deliver as much information as possible when data is being aggregated and published as LOD.

[19] http://dbpedia.org

Persons

Identifying properties:

- name;
- date of birth;
- place of birth.

```
PREFIX dbpediaprop: < http://live.dbpedia.org/property/dateOfBirth>
PREFIX dbpedia: <http://dbpedia.org/resource/ >
PREFIX dbpediaowl: <http://live.dbpedia.org/ontology/>
PREFIX foaf: < http://xmlns.com/foaf/0.1/>
SELECT ?person
WHERE
{
?person a foaf:Person.
?person dbpediaprop:name 'Gerhard Richter@en.
?person dbpediaprop:dateOfBirth '1932-02-09'^^xsd:date.
?person dbpediaowl:birthPlace dbpedia:Dresden.
}

Result: <http://dbpedia.org/resource/Gerhard_Richter>
```

Listing 4: example SPARQL query for DBpedia looking for a Gerhard Richter entity

The same holds true here, as with the places. If not all identifying properties are present in the record, the query will become less distinctive and more than one result could return. The solution is to have someone acting as an authority, picking out the right enrichment.

Artwork type

Identifying properties:

- The literal itself

The artwork type should actually refer to a term of a SKOS vocabulary. The SPARQL query below shows how to link the artwork type to a SKOS concept.

```
PREFIX skos: < http://xmlns.com/foaf/0.1/>
SELECT ?concept
WHERE
{
?concept a skos:Concept.
?concept skos:prefLabel 'Video Art'@en.
}
```

Listing 5: example SPARQL query looking for a concept named "Video Art"

Artwork

Identifying properties:

- Artwork ID

- Data provider

The artwork itself can be enriched with its *Europeana* equivalent, of course only if the artwork's metadata has already been delivered to *Europeana*. A combination of the ID and the data provider is sufficient to identify the *Europeana* equivalent. If *Europeana* supported SPARQL, the query would look like this:

```
PREFIX skos: < http://xmlns.com/foaf/0.1/>
SELECT ?artwork
WHERE
{
?artwork a ore:Aggregation.
?artwork edm:aggregatedCHO 'Hirst:1423'
?artwork edm:dataProvider 'MuZee'.
}
```

Listing 6: example SPARQL query for Europeana looking for an artwork of Hirst

In our PoC, the enrichment module is implemented in Java. It will query remote data sources looking for enrichments for each sort of entity (artwork, person, location, etc.). When querying the remote dataset, the identifying properties of the entities are used. Of course, not every DCA artwork description will contain all the information on identifying properties. In these cases, the query will be weakened using only the provided identifying properties of the entity. Such automatic enrichments are always subjected to many errors and human intervention is needed to ensure the found enrichment is correct. To solve this, the provenance is stored for each enrichment. This way, enrichments can be classified based on the use of identifying properties. Those using all the identifying properties are correct, those using only a subset of the identifying properties will need inspection to see if the found enrichment is still correct.

5. Enrichment sources

Purpose

In this section, we will elaborate on data sources that can be used to enrich one's data. These sources either make their data searchable via a SPARQL endpoint or via an API. These enrichment sources are classified based on the topic of their content.

Named entities

Named Entities are just known concepts, such as important events, places, persons, etc. Usually, named entities are extracted from free text. Once you have selected the named entities from a text, you can start using the other data sources to look for enrichments for them.

OpenCalais[20]

OpenCalais is the most famous named entity extractor. You can send text to OpenCalais and it will filter all the recognised named entities from the text. It can detect persons, locations, events, brands, music bands, etc. OpenCalais immediately links to DBpedia, Wikipedia, Freebase, Reuters.com, GeoNames, Shopping.com, IMDB, and Linked MDB. It provides web services which automatically annotate your content with rich semantic metadata. As well as its web services, OpenCalais also publishes its content as LOD. OpenCalais can only deal in English, French and Spanish.

DBpedia Spotlight[21]

DBpedia Spotlight is a named entity extractor like OpenCalais. It will annotate your free text with links to DBpedia resources. One drawback is that it is currently only available in English.

Cross-domain

These cross-domain datasets have information on various sorts of things, e.g., persons, locations, movies, books, cultural heritage, etc. They can almost always be used to enrich your data, no matter what domain your data belongs to. When looking at the giant global data graph, you will notice that these datasets are mostly used as a sort of interlinking hubs, which connect various other datasets to each other.

DBPedia[22]

DBpedia publishes the knowledge extracted from Wikipedia as LOD. It makes its data available via a SPARQL endpoint. It has information on persons, places, creative works (music albums, films, video games, etc.), organisations, species and diseases. DBpedia provides localised versions of DBpedia in 111 languages. It publishes its content as LOD and has a public SPARQL endpoint available for querying the dataset. DBpedia also provides RDF dumps, which can be downloaded and ingested into triple stores.

Freebase[23]

Freebase is very similar to DBpedia. It holds information on persons, locations, organisations, etc. The data from Freebase is collected from various data sources, such as ArXiv, CrunchBase, Eurostat, Wikipedia, IMDB, Library of Congress, etc. Freebase can take on queries using MQL, their

[20] http://www.opencalais.com/
[21] http://github.com/dbpedia-spotlight/dbpedia-spotlight
[22] http://dbpedia.org
[23] http://www.freebase.com/

own developed query language. They also provide a REST API and publish the content also as LOD. RDF dumps of Freebase are also available.

OpenCyc[24]

OpenCyc is the open source version of CYC. CYC is a very large general knowledge base (including a reasoning engine). The CYC ontology contains hundreds of thousands of terms and millions of assertions, relating the terms to each other, forming an (English) upper ontology of which all of human consensus reality domain is the domain. OpenCyc provides an API for application development.

YAGO2[25]

YAGO2 is a large semantic knowledge base derived from Wikipedia, WordNet, and GeoNames. It contains knowledge on 10 million entities (persons, organisations, cities, etc.). YAGO2 provides dumps, but it can also be queried and browsed.

Vocabularies

WordNet[26]

WordNet is a large lexical database of English. It organises nouns, verbs, adjectives, and adverbs in sets of cognitive synonyms (synsets), each expressing a distinct concept. WordNet superficially resembles a controlled vocabulary. WordNet can be browsed and is also available as dump.

LCSH[27]

Library of Congress Subject Headings are a thesaurus of subject headings. These subject headings capture the essence of the topic of a document. The thesaurus can be used as controlled vocabulary to classify bibliographic records. LCSH are currently available as LOD, as a SKOS vocabulary and as an RDF document, but they do not provide a SPARQL endpoint. The RDF documents can be downloaded and stored in a local triple store to query the vocabularies using SPARQL.

Cultural Heritage

Europeana[28]

Europeana is a single access point to millions of books, paintings, films, museum objects and archival records that have been digitised throughout Europe. It is an authoritative source of information coming from European cultural and scientific institutions. The content from *Europeana* is available via an API. The *Europeana* LOD pilot, also makes a part of the content available as LOD.

The Data Hub[29]

The Data Hub contains 4.293 datasets that one can browse, learn about and download. In this data hub there are many datasets focusing on art. One can search for the dataset one needs and download it to use as an enrichment source. Examples of interesting datasets for the art sector are:

http://thedatahub.org/group/open-glam

http://thedatahub.org/group/art

[24] http://www.opencyc.org/
[25] http://www.mpi-inf.mpg.de/yago-naga/yago/
[26] http://wordnet.princeton.edu/
[27] http://id.loc.gov/authorities/subjects.html
[28] http://www.europeana.eu
[29] http://thedatahub.org/

http://thedatahub.org/dataset/freebase-visual-art

http://thedatahub.org/dataset/grants-for-the-arts-awards-arts-council-england

http://thedatahub.org/dataset/ukgac

Locations

GeoNames[30]

GeoNames is a geographical database, with information on all countries. It contains over 8 million place names. The data is available as a dump, or via web services.

World Factbook[31]

The CIA World Factbook provides information on the history, people, government, economy, geography, communications, transportation, military and transnational issues of 267 world entities. The World Factbook is available as dump to download.

Persons

VIAF[32]

The Virtual International Authority File initiative is a joint project of several national libraries plus selected regional and trans-national library agencies. The project's goal is to lower the cost and increase the utility of library authority files by matching and linking widely-used authority files and making that information available on the Web. This data source can be used to enrich persons, in particular artists.

Movies

Linked MDB[33]

The Linked Movie DataBase publishes LOD on movies. The Linked MDB also links to DBpedia, YAGO, Flickr wrappr, RDF Book Mashup, MusicBrainz, GeoNames, IMDB, Rotten Tomatoes and Freebase. The content is thus published as LOD and a SPARQL endpoint is made available for querying the dataset.

Music

Music Brainz[34]

MusicBrainz is a music knowledge base. It contains information on artists, release groups, releases, recordings, works, and labels, as well as relationships between them. MusicBrainz data is free to download and is also offered in RDF for download.

BBC Music[35]

BBC Music is also a music knowledge base. It contains information on all the artists ever played at the BBC or in one of their radio shows. It gathers also information from MusicBrainz and Wikipedia. The data is available via LOD, and web services.

[30] http://www.geonames.org/
[31] https://www.cia.gov/library/publications/the-world-factbook/
[32] http://viaf.org/
[33] http://www.linkedmdb.org/
[34] http://musicbrainz.org/
[35] http://www.bbc.co.uk/music

Books

RDF Book Mashup[36]

The RDF Book Mashup makes information available about books, their authors, reviews, and online bookstores. The data is taken from data sources like Amazon, Google, and Yahoo. The information is published on the Web as LOD and the data can be queried using SPARQL.

Proof of concept, developed within the framework of DCA

Our enrichment module will enrich the following entities, as shown in the previous section:

- artwork;
- persons;
- locations;
- artwork type.

For each type, we need to select enrichment sources to search for possible enrichments.

Named Entity Extraction

Before enriching the specified entities, we must first extract the named entities from the EDM instances using OpenCalais. Those to be extracted are: locations, persons, and organisations.

Enrichment Sources Artwork

The artwork, published as LOD, will be linked to its *Europeana* equivalent. Of course, it will need to be published already for *Europeana* to do the enrichment. Another source for artwork enrichment is Freebase, more particular its visual arts collection.

- *Europeana*
- Freebase Visual Art[37]

Enrichment Sources Persons

The actors in the EDM instances will be linked to persons described in DBpedia, Freebase and VIAF. DBpedia and Freebase both serve as enrichment hubs and have extensive information on enrichments for their resources. VIAF, which offers authority records on artists, is used as another enrichment source for persons, in particular artists.

- DBpedia
- Freebase
- VIAF

Enrichment Sources Locations

The locations detected in the EDM instances will be enriched using GeoNames and DBpedia. GeoNames is especially targeted for describing locations. DBpedia is cross-domain, and also has a lot of information on locations.

- GeoNames

[36] http://www4.wiwiss.fu-berlin.de/bizer/bookmashup/
[37] http://www.freebase.com/view/visual_art

- DBpedia

Enrichment Sources Artwork Types

The literals denoting an artwork type will be replaced by a SKOS concept from the DCA vocabulary, discussed in detail in deliverable 3.1 *Metadata implementation guidelines for digitised contemporary artworks* and deliverable 3.2 *Recommendation on contextualisation and interlinking digitised contemporary artworks*.

- DCA SKOS vocabulary

6. LOD strategies for DCA partners

In this section, we provide feasible strategies for DCA partners to publish their content as LOD and to make it possible to interlink all those published entities. The whole workflow has already been discussed and consists of the following steps:

- select appropriate RDF model to publish your data;
- choose an LOD server infrastructure;
- transform your data to RDF;
- enrich your data.

Select appropriate RDF model to publish your data

The DCA partners are institutions on contemporary art. They all hold valuable heritage information. The best model with which to publish this content as LOD is EDM. EDM is geared towards LOD publication of heritage data. At the same time, EDM and LIDO are closely related and the DCA partners have all already mapped their content to LIDO and are therefore familiar with LIDO.

Choose an LOD server infrastructure

The next thing is to choose your LOD infrastructure. For this, the DCA partners have many options, depending on their ICT knowledge and the dynamism of their data.

Organisations with no ICT staff should choose a solution that offers everything to publish the data as LOD. Solutions here are Openlink Virtuoso or D2R Server. Both implement a Linked Data frontend and a SPARQL endpoint over HTTP. Organisations with ICT staff could compile their own solution for LOD publication, just as in our proof of concept. Good triple stores are Openlink Virtuoso and TDB. If you choose Openlink Virtuoso, the Linked Data frontend and SPARQL endpoint are already in place. Otherwise, Pubby is a good solution for making the Linked Data frontend. Joseki or Fuseki are good solutions for providing a SPARQL endpoint over HTTP.

If your data changes a lot and is situated in a relational database, D2R server or Virtuoso Triplify are good solutions. They create an extra layer over your relational database, relying on a JDBC connection to the database. The benefit of this approach is that your data is transformed to RDF on-the-fly. This avoids having syncing problems between the triple store and your relational database. The other benefit of this approach is that the partner’s business processes are not affected. They all can just keep on doing their job on the relational database. If you migrate from a relational database to a triple store, these processes will need to be adapted to communicating with it instead of the relational database. From the DCA partner survey it seems that most of them are using a relational database, and that this is the preferred solution. A tutorial on D2R Server will be given to facilitate the process during the last F2F meeting of the DCA partner.

Transform your data to RDF (and ingest)

If you use D2R server or Virtuoso Triplify, you just need to configure the applications to publish your data as RDF. If you migrate all your data to a triple store, you will first need to transform it. This can be done using XSLT or via RDFisers. Many XSLT and RDFisers are available on the Web and free to use for data conversion. Once this is done, the generated RDF data can be ingested into the triple store.

Enrich your data

The only tool used at the moment for enriching data is SILK. A preliminary to using SILK is knowledge of SPARQL. one disadvantage of using SILK is that only data sources can be contacted providing a SPARQL endpoint over HTTP. Another strategy could be to upload your data to *Europeana* and let them enrich it for you. Then *Europeana* can give you back the generated enrichments from *Europeana.* So you could use *Europeana* as an enrichment source.

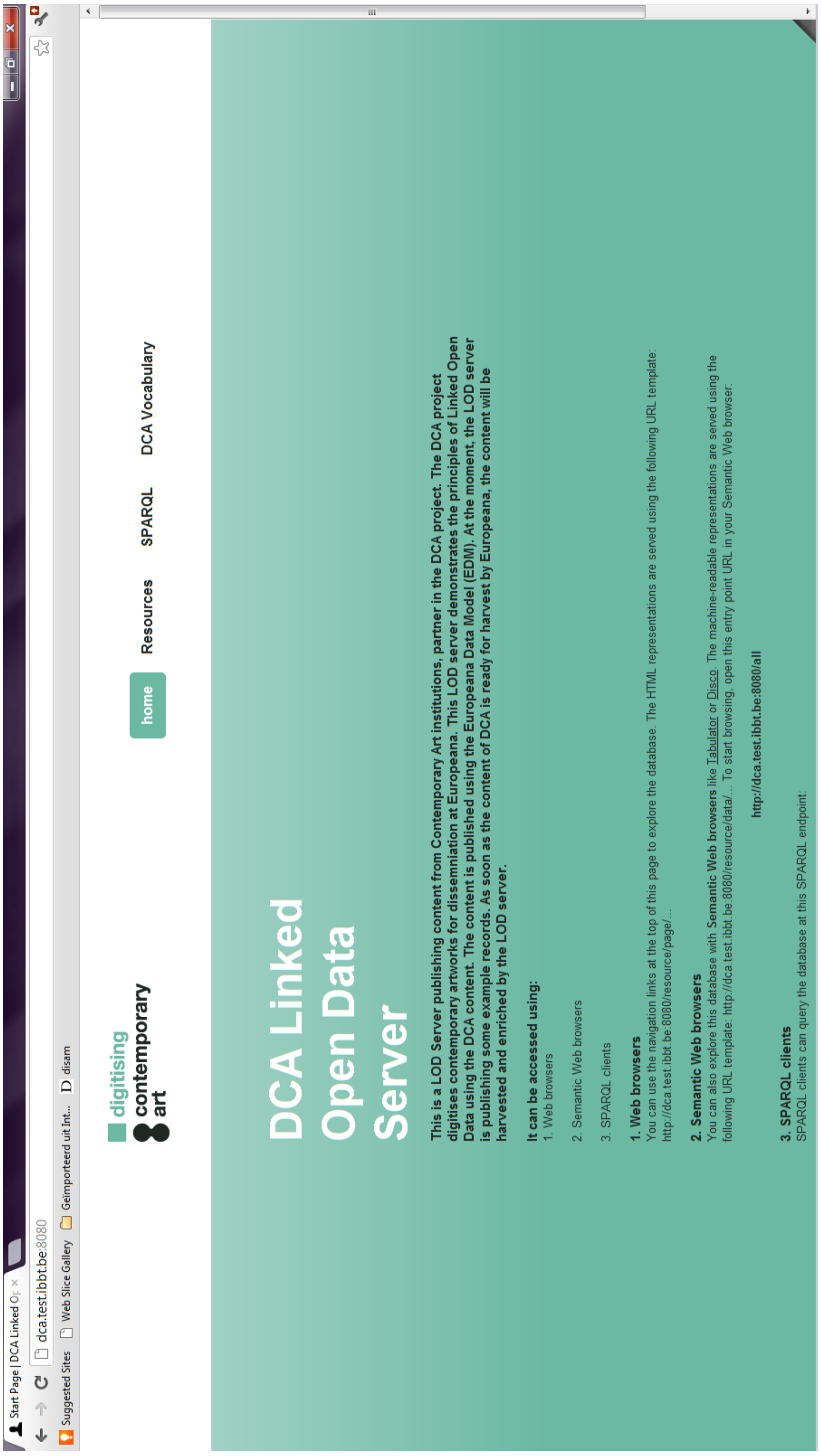
You could also develop your own enrichment module. When doing so, it is good practice to start with OpenCalais to extract the named entities and then use them to look for enrichments in DBpedia. Because DBpedia serves as an enrichment hub, a lot can just be taken directly from it.

7. Bibliography

1. *Linked Data - The Story so far.* Christian Bizer, Tom Heath, and Tim Berners-Lee. 3, s.l. : International Journal on Semantic Web and Information, 2009, Vol. 5.
2. *RDF Primer.* [Online] http://www.w3.org/TR/rdf-primer/.
3. *HTML Specification.* [Online] http://www.w3.org/TR/html401/.
4. *RDF/XML Syntax Specification (Revised).* [Online] http://www.w3.org/TR/rdf-syntax-grammar/.
5. *Turtle - Terse RDF Triple Language.* [Online] http://www.w3.org/TeamSubmission/turtle/.
6. *RDF Vocabulary Description Language 1.0: RDF Schema.* [Online] http://www.w3.org/TR/rdf-schema/.
7. *OWL Web Ontology Language .* [Online] http://www.w3.org/TR/owl-ref/.
8. Dublin Core Metadata Initiative Homepage. [Online] http://dublincore.org/.
9. *Ontology:DOLCE+DnS Ultralite.* s.l. : ontologydesignpatterns.org, 2006.
10. Dan Brickley, Libby Miller. *FOAF Vocabulary Specification 0.98.* 2010.
11. Brickley, Dan. *Basic Geo (WGS84 lat/long) Vocabulary.* s.l. : W3C, 2004.
12. Ryan Shaw, Raphaël Troncy, Lynda Hardman. *LODE: An ontology for Linking Open Descriptions of Events.* 2010.
13. *Europeana Data Model.* [Online] http://group.Europeana.eu/c/document_library/get_file?uuid=718a3828-6468-4e94-a9e7-7945c55eec65&groupId=10605.
14. Dow, Martin. *The OpenART Linked Events Model.* 2010.
15. Hepp, Martin. *GoodRelations Language Reference.* 2011.
16. Carl Lagoze, Herbert Van de Sompel, Michael Nelson, Simeon Warner. *The Open Archives Initiative Protocol for Metadata Harvesting.* 2008.
17. *LIDO - Lightweight Information Describing Objects.* [Online] http://www.lido-schema.org/schema/v1.0/lido-v1.0-specification.pdf.
18. Kay, Michael. *XSL Transformations (XSLT) Version 2.0.* 2007.
19. Antoine Isaac, Ed Summers. *SKOS Simple Knowledge Organization System Primer.* 2009.
20. Ben Adida, Ivan Herman, Manu Sporny, Mark Birbeck. *RDFa 1.1 Primer.* 2012.
21. Eric Prud'hommeaux, Andy Seaborne. *SPARQL Query Language for RDF.* 2007.
22. *SEPIA Data Element Set (SEPIADES).* [Online] http://marinemetadata.org/references/sepiades.
23. *Metadata Encoding and Transmission Standard.* [Online] 2009 йил 29-01. http://www.loc.gov/standards/mets/.
24. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* [Online] http://www.w3.org/TR/REC-xml/.
25. *XML Schema.* [Online] http://www.w3.org/XML/Schema.
26. *Anglo-American Cataloguing Rules.* [Online] http://www.aacr2.org/.
27. *Resource Description and Access.* [Online] http://www.rda-jsc.org/rda.html.
28. International standard bibliographic description (ISBD). [Online] http://archive.ifla.org/VII/s13/pubs/ISBD_consolidated_2007.pdf.

29. *MARC 21 Specifications for Record Structure, Character Sets, and Exchange Media.* [Online] http://www.loc.gov/marc/specifications/.
30. *FUNCTIONAL REQUIREMENTS FOR BIBLIOGRAPHIC RECORDS.* [Online] http://archive.ifla.org/VII/s13/frbr/frbr_current_toc.htm.
31. *Definition of the CIDOC Conceptual Reference Model.* [Online] http://www.cidoc-crm.org/docs/cidoc_crm_version_5.0.3.pdf.
32. *Categories for the Descriptions of Works of Art.* [Online] http://www.getty.edu/research/publications/electronic_publications/cdwa/index.html.
33. *SPECTRUM, the standard for collections management.* [Online] http://www.collectionstrust.org.uk/spectrum.
34. *General International Standard Archival.* [Online] http://www.icacds.org.uk/eng/ISAD(G).pdf.
35. *Encoded Archival Description.* [Online] http://www.loc.gov/ead/.
36. *Art & Architecture Thesaurus.* [Online] http://www.getty.edu/research/tools/vocabularies/aat/about.html.
37. *Getty Thesaurus of Geographic Names.* [Online] http://www.getty.edu/research/tools/vocabularies/tgn/about.html.
38. *Union List of Artist Names.* [Online] http://www.getty.edu/research/tools/vocabularies/ulan/about.html.
39. *Library of Congress Subject Headings.* [Online] http://id.loc.gov/authorities/subjects.html.
40. *RKDartists&.* [Online] http://website.rkd.nl/Databases/RKDartists.
41. *Film identification - Enhancing interoperability of metadata - Element sets and structures.* [Online] http://filmstandards.org/fsc/index.php/EN_15907.
42. *Open Archives Initiative: Object Reuse and Exchange.* [Online] http://www.openarchives.org/ore/1.0/primer.html.
43. *Europeana Semantic Elements Specification.* [Online] http://version1.Europeana.eu/c/document_library/get_file?uuid=77376831-67cf-4cff-a7a2-7718388eec1d&groupId=10128.
44. *Reference Model for an Open Archival Information System.* [Online] http://public.ccsds.org/publications/archive/650x0b1.pdf.
45. *PREMIS Data Dictionary for Preservation Metadata.* [Online] http://www.loc.gov/standards/premis/v2/premis-dd-2-1.pdf.
46. Europeana. *Definition of the Europeana Data.* 2012.

8. Appendix: screenshots of the DCA LOD server



DCA Linked Open Data Server

This is a LOD Server publishing content from Contemporary Art institutions, partner in the DCA project. The DCA project digitises contemporary artworks for dissemniation at Europeana. This LOD server demonstrates the principles of Linked Open Data using the DCA content. The content is published using the Europeana Data Model (EDM). At the moment, the LOD server is publishing some example records. As soon as the content of DCA is ready for harvest by Europeana, the content will be harvested and enriched by the LOD server.

It can be accessed using:

1. Web browsers
2. Semantic Web browsers
3. SPARQL clients

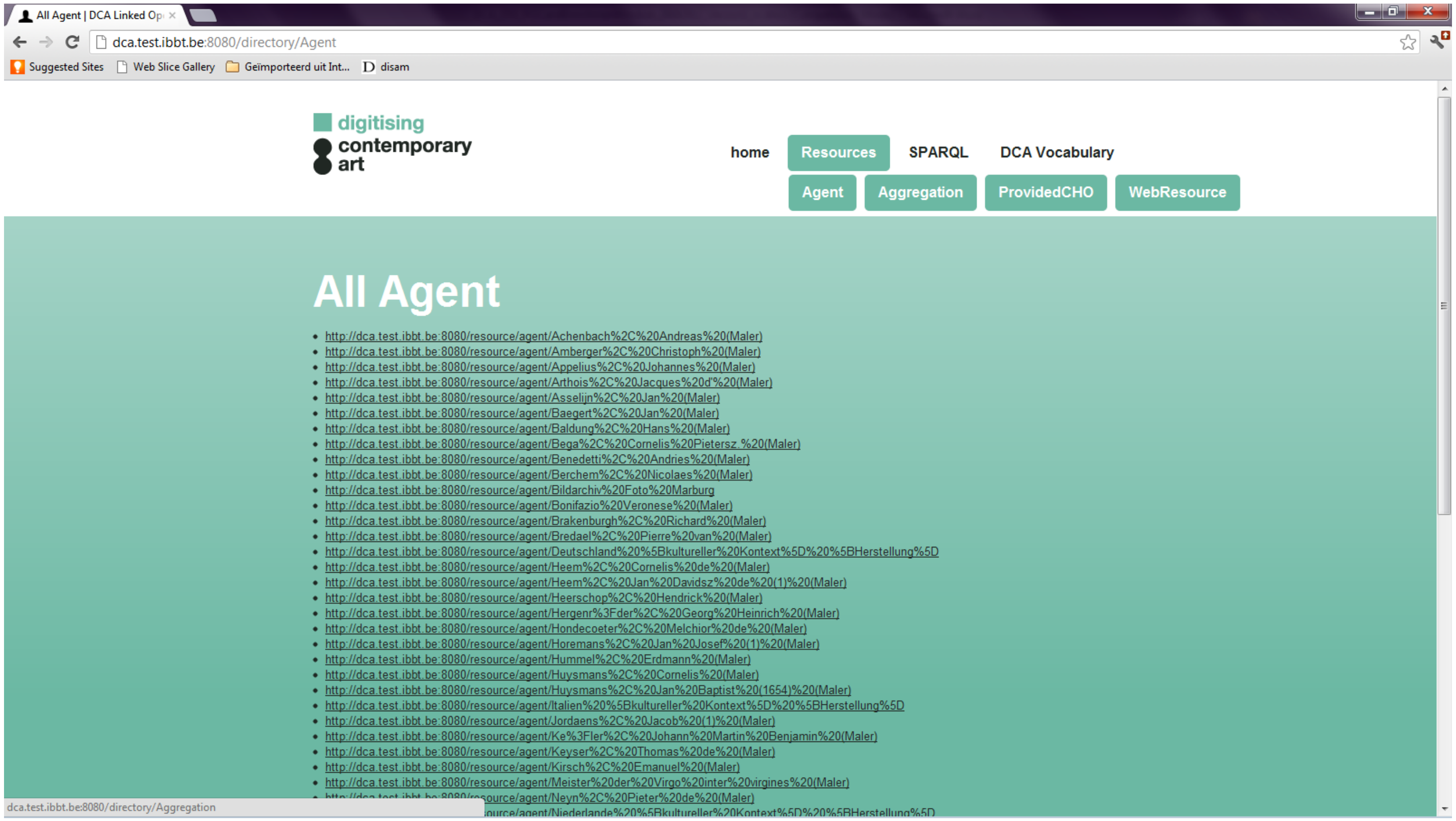
1. Web browsers
You can use the navigation links at the top of this page to explore the database. The HTML representations are served using the following URL template: http://dca.test.ibbt.be:8080/resource/page/...

2. Semantic Web browsers
You can also explore this database with Semantic Web browsers like Tabulator or Disco. The machine-readable representations are served using the following URL template: http://dca.test.ibbt.be:8080/resource/data/... To start browsing, open this entry point URL in your Semantic Web browser:

http://dca.test.ibbt.be:8080/all

3. SPARQL clients
SPARQL clients can query the database at this SPARQL endpoint:

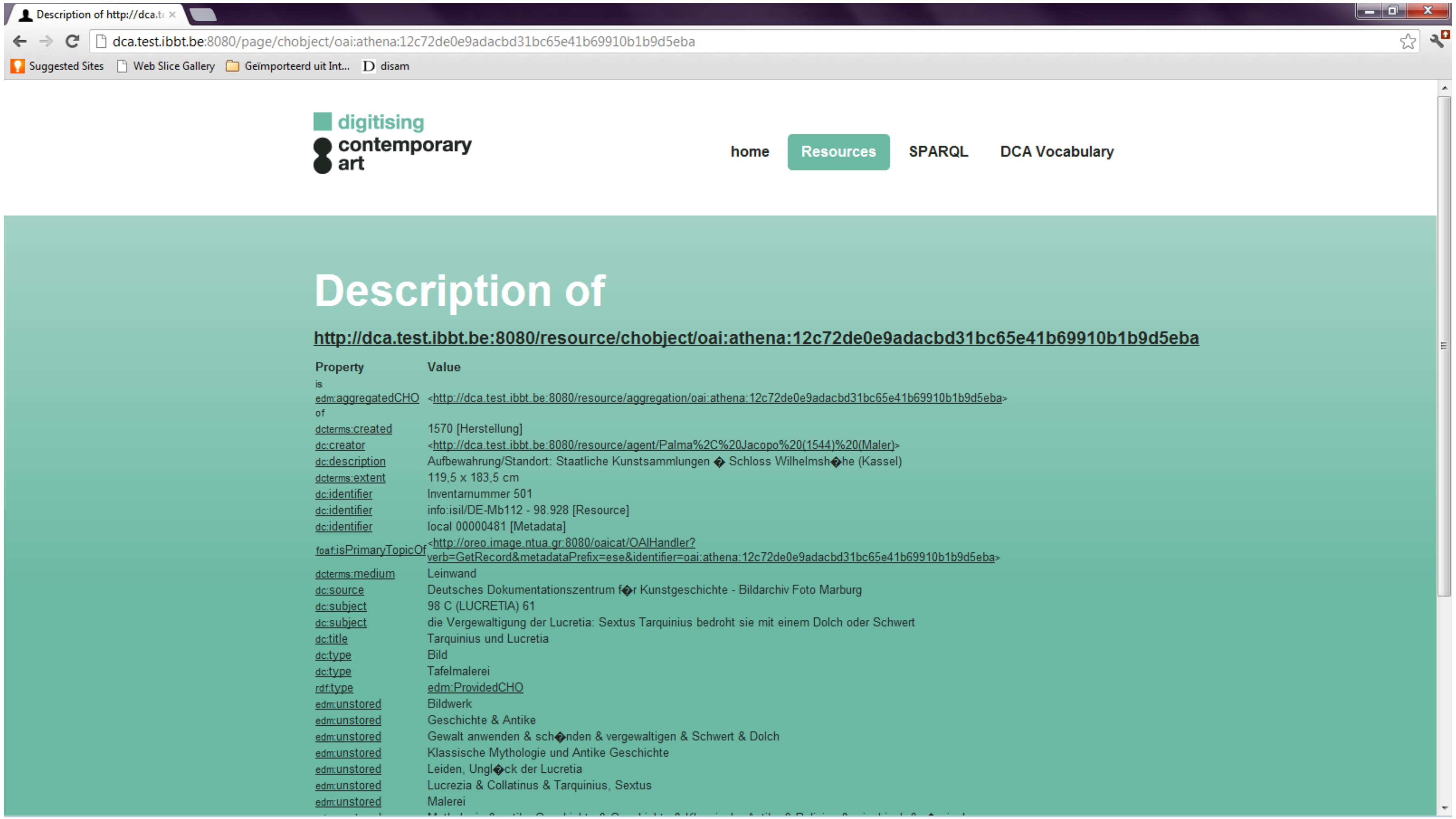
Figure 5: screenshot of the homepage of our PoC


All Agent | DCA Linked Op
dca.test.ibbt.be:8080/directory/Agent
Suggested Sites
Web Slice Gallery
Geïmporteerd uit Int...
disam
digitising contemporary art
home
Resources
SPARQL
DCA Vocabulary
Agent
Aggregation
ProvidedCHO
WebResource
dca.test.ibbt.be:8080/directory/Aggregation

All Agent

- http://dca.test.ibbt.be:8080/resource/agent/Achenbach%2C%20Andreas%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Amberger%2C%20Christoph%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Appelius%2C%20Johannes%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Arthois%2C%20Jacques%20d'%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Asselijn%2C%20Jan%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Baegert%2C%20Jan%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Baldung%2C%20Hans%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Bega%2C%20Cornelis%20Pietersz.%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Benedetti%2C%20Andries%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Berchem%2C%20Nicolaes%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Bildarchiv%20Foto%20Marburg
- http://dca.test.ibbt.be:8080/resource/agent/Bonifazio%20Veronese%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Brakenburgh%2C%20Richard%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Bredael%2C%20Pierre%20van%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Deutschland%20%5Bkultureller%20Kontext%5D%20%5BHerstellung%5D
- http://dca.test.ibbt.be:8080/resource/agent/Heem%2C%20Cornelis%20de%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Heem%2C%20Jan%20Davidsz%20de%20(1)%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Heerschop%2C%20Hendrick%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Hergenr%3Fder%2C%20Georg%20Heinrich%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Hondecoeter%2C%20Melchior%20de%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Horemans%2C%20Jan%20Josef%20(1)%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Hummel%2C%20Erdmann%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Huysmans%2C%20Cornelis%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Huysmans%2C%20Jan%20Baptist%20(1654)%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Italien%20%5Bkultureller%20Kontext%5D%20%5BHerstellung%5D
- http://dca.test.ibbt.be:8080/resource/agent/Jordaens%2C%20Jacob%20(1)%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Ke%3Fler%2C%20Johann%20Martin%20Benjamin%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Keyser%2C%20Thomas%20de%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Kirsch%2C%20Emanuel%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Meister%20der%20Virgo%20inter%20virgines%20(Maler)
- http://dca.test.ibbt.be:8080/resource/agent/Neyn%2C%20Pieter%20de%20(Maler)
- source/agent/Niederlande%20%5Bkultureller%20Kontext%5D%20%5BHerstellung%5D

Figure 6: screenshot of webpage listing all edm:Agent entities of the PoC

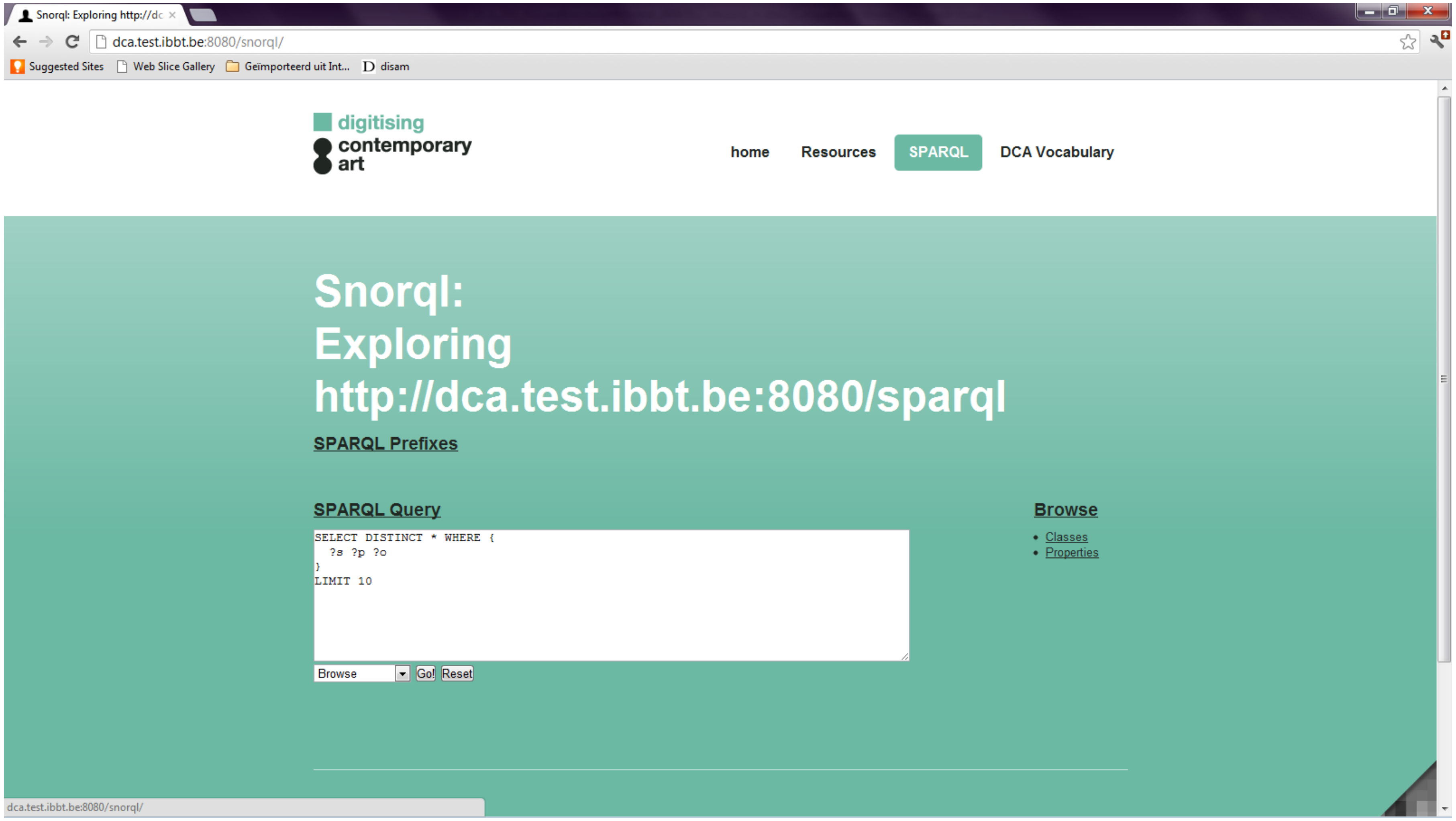

Description of http://dca.t
dca.test.ibbt.be:8080/page/chobject/oai:athena:12c72de0e9adacbd31bc65e41b69910b1b9d5eba
Suggested Sites
Web Slice Gallery
Geïmporteerd uit Int...
disam
digitising contemporary art
home
Resources
SPARQL
DCA Vocabulary

Description of

http://dca.test.ibbt.be:8080/resource/chobject/oai:athena:12c72de0e9adacbd31bc65e41b69910b1b9d5eba

Property	Value
is edm:aggregatedCHO of	<http://dca.test.ibbt.be:8080/resource/aggregation/oai:athena:12c72de0e9adacbd31bc65e41b69910b1b9d5eba>
dcterms:created	1570 [Herstellung]
dc:creator	<http://dca.test.ibbt.be:8080/resource/agent/Palma%2C%20Jacopo%20(1544)%20(Maler)>
dc:description	Aufbewahrung/Standort: Staatliche Kunstsammlungen � Schloss Wilhelmsh�he (Kassel)
dcterms:extent	119,5 x 183,5 cm
dc:identifier	Inventarnummer 501
dc:identifier	info:isil/DE-Mb112 - 98.928 [Resource]
dc:identifier	local 00000481 [Metadata]
foaf:isPrimaryTopicOf	<http://oreo.image.ntua.gr:8080/oaicat/OAIHandler?verb=GetRecord&metadataPrefix=ese&identifier=oai:athena:12c72de0e9adacbd31bc65e41b69910b1b9d5eba>
dcterms:medium	Leinwand
dc:source	Deutsches Dokumentationszentrum f�r Kunstgeschichte - Bildarchiv Foto Marburg
dc:subject	98 C (LUCRETIA) 61
dc:subject	die Vergewaltigung der Lucretia: Sextus Tarquinius bedroht sie mit einem Dolch oder Schwert
dc:title	Tarquinius und Lucretia
dc:type	Bild
dc:type	Tafelmalerei
rdf:type	edm:ProvidedCHO
edm:unstored	Bildwerk
edm:unstored	Geschichte & Antike
edm:unstored	Gewalt anwenden & sch�nden & vergewaltigen & Schwert & Dolch
edm:unstored	Klassische Mythologie und Antike Geschichte
edm:unstored	Leiden, Ungl�ck der Lucretia
edm:unstored	Lucrezia & Collatinus & Tarquinius, Sextus
edm:unstored	Malerei

Figure 7: screenshot showing the description of an edm:ProvidedCHO instance of the PoC


Snorql: Exploring http://dc
dca.test.ibbt.be:8080/snorql/
Suggested Sites
Web Slice Gallery
Geïmporteerd uit Int...
disam
digitising contemporary art
home
Resources
SPARQL
DCA Vocabulary

Snorql: Exploring http://dca.test.ibbt.be:8080/sparql

SPARQL Prefixes

SPARQL Query

```
SELECT DISTINCT * WHERE {
  ?s ?p ?o
}
LIMIT 10
```

Browse

- Classes
- Properties

dca.test.ibbt.be:8080/snorql/

Figure 8: screenshot of the SPARQL endpoint interface of PoC