How to Publish Linked Data on the Web

Authors:

Chris Bizer (Web-based Systems Group, Freie Universität Berlin, Germany)

Richard Cyganiak (Web-based Systems Group, Freie Universität Berlin, Germany)

Tom Heath (Knowledge Media Institute, The Open University, Milton Keynes, UK)

This version:

http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/20070727/

Latest version:

http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/

Abstract

This document provides a tutorial on how to publish Linked Data on the Web. After a general overview of the concept of Linked Data, we describe several practical recipes for publishing information as Linked Data on the Web.

This tutorial has been superseeded by the book Linked Data: Evolving the Web into a Global Data Space written by Tom Heath and Christian Bizer. This tutorial was published in 2007 and is still online for historical reasons. The Linked Data book was published in 2011 and provides a more detailed and up-to-date introduction into Linked Data.

Table of Contents

---

1. Introduction: Linked Data on the Web

The goal of Linked Data is to enable people to share structured data on the Web as easily as they can share documents today.

The term Linked Data was coined by Tim Berners-Lee in his Linked Data Web architecture note. The term refers to a style of publishing and interlinking structured data on the Web. The basic assumption behind Linked Data is that the value and usefulness of data increases the more it is interlinked with other data. In summary, Linked Data is simply about using the Web to create typed links between data from different sources.

The basic tenets of Linked Data are to:

1. use the RDF data model to publish structured data on the Web
2. use RDF links to interlink data from different data sources

Applying both principles leads to the creation of a data commons on the Web, a space where people and organizations can post and consume data about anything. This data commons is often called the Web of Data or Semantic Web.

The Web of Data can be accessed using Linked Data browsers, just as the traditional Web of documents is accessed using HTML browsers. However, instead of following links between HTML

pages, Linked Data browsers enable users to navigate between different data sources by following RDF links. This allows the user to start with one data source and then move through a potentially endless Web of data sources connected by RDF links. For instance, while looking at data about a person from one source, a user might be interested in information about the person's home town. By following an RDF link, the user can navigate to information about that town contained in another dataset.

Just as the traditional document Web can be crawled by following hypertext links, the Web of Data can be crawled by following RDF links. Working on the crawled data, search engines can provide sophisticated query capabilities, similar to those provided by conventional relational databases. Because the query results themselves are structured data, not just links to HTML pages, they can be immediately processed, thus enabling a new class of applications based on the Web of Data.

The glue that holds together the traditional document Web is the hypertext links between HTML pages. The glue of the data web is RDF links. An RDF link simply states that one piece of data has some kind of relationship to another piece of data. These relationships can have different types. For instance, an RDF link that connects data about people can state that two people know each other; an RDF link that connects information about a person with information about publications in a bibliographic database might state that a person is the author of a specific paper.

There is already a lot of structured data accessible on the Web through Web 2.0 APIs such as the eBay, Amazon, Yahoo, and Google Base APIs. Compared to these APIs, Linked Data has the advantage of providing a single, standardized access mechanism instead of relying on diverse interfaces and result formats. This allows data sources to be:

- more easily crawled by search engines,
- accessed using generic data browsers, and
- enables links between data from different data sources.

Having provided a background to Linked Data concepts, the rest of this document is structured as follows: Section 2 outlines the basic principles of Linked Data. Section 3 provides practical advice on how to name resources with URI references. Section 4 discusses terms from well-known vocabularies and data sources which should be reused to represent information. Section 5 explains what information should be included into RDF descriptions that are published on the Web. Section 6 gives practical advice on how to generate RDF links between data from different data sources. Section 7 presents several complete recipes for publishing different types of information as Linked Data on the Web using existing Linked Data publishing tools. Section 8 discusses testing and debugging Linked Data sources. Finally Section 9 gives an overview of alternative discovery mechanisms for Linked Data on the Web.

2. Basic Principles

This chapter describes the basic principles of Linked Data. As Linked Data is closely aligned to the general architecture of the Web, we first summarize the basic principles of this architecture. Then we give an overview of the RDF data model and recommend how the data model should be used in the Linked Data context.

2.1. Web Architecture

This section summarizes the basic principles of the Web Architecture and introduces terminology such as *resource* and *representation*. For more detailed information please refer to the Architecture of the World Wide Web, Volume One W3C recommendation and the current findings of the W3C Technical Architecture Group (TAG).

Resources

To publish data on the Web, we first have to *identify the items of interest* in our domain. They are the things whose properties and relationships we want to describe in the data. In Web Architecture terminology, all items of interest are called *resources*.

In 'Dereferencing HTTP URIs' the W3C Technical Architecture Group (TAG) distinguish between two kinds of resources: *information resources* and *non-information resources* (also called 'other resources') . This distinction is quite important in a Linked Data context. All the resources we find on the traditional document Web, such as documents, images, and other media files, are information resources. But many of the things we want to share data about are not: People, physical products, places, proteins, scientific concepts, and so on. As a rule of thumb, all “real-world objects” that exist outside of the Web are non-information resources.

Resource Identifiers

Resources are identified using *Uniform Resource Identifiers (URIs)*. In the context of Linked Data, we restrict ourselves to using HTTP URIs only and avoid other URI schemes such as URNs and DOIs. HTTP URIs make good names for two reasons: They provide a simple way to create globally unique names without centralized management; and URIs work not just as a name but also as a means of accessing information about a resource over the Web. The preference for HTTP over other URI schemes is discussed at length in the W3C TAG draft finding URNs, Namespaces and

Registries.

Representations

Information resources can have *representations*. A representation is a stream of bytes in a certain format, such as HTML, RDF/XML, or JPEG. For example, an invoice is an information resource. It could be represented as an HTML page, as a printable PDF document, or as an RDF document. A single information resource can have many different representations, e.g. in different formats, resolution qualities, or natural languages.

Dereferencing HTTP URIs

URI Dereferencing is the process of looking up a URI on the Web in order to get information about the referenced resource. The W3C TAG draft finding about Dereferencing HTTP URIs introduced a distinction on how URIs identifying information resources and non-information resources are dereferenced:

- *Information Resources*: When a URI identifying an information resource is dereferenced, the server of the URI owner usually generates a new representation, a new snapshot of the information resource's current state, and sends it back to the client using the HTTP response code `200 OK`.
- *Non-Information Resources* cannot be dereferenced directly. Therefore Web architecture uses a trick to enable URIs identifying non-information resources to be dereferenced: Instead of sending a representation of the resource, the server sends the client the URI of a information resource which describes the non-information resource using the HTTP response code `303 See Other`. This is called a 303 redirect. In a second step, the client dereferences this new URI and gets a representation describing the original non-information resource.

Note: There are two approaches that data publishers can use to provide clients with URIs of information resources describing non-information resources: Hash URIs and 303 redirects. This document focuses mostly on the 303 redirect approach. See Section 4.3 of *Cool URIs for the Semantic Web* for a discussion of the tradeoffs between both approaches.

Content Negotiation

HTML browsers usually display RDF representations as raw RDF code, or simply download them as RDF files without displaying them. This is not very helpful to the average user. Therefore, serving a proper HTML representation in addition to the RDF representation of a resource helps humans to

figure out what a URI refers to.

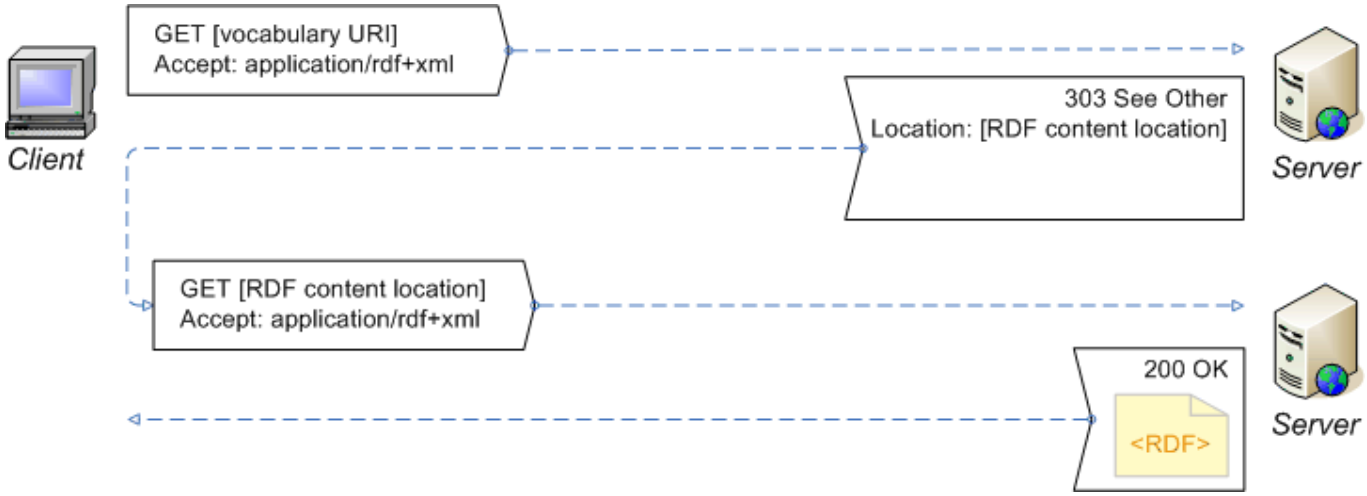
This can be achieved using an HTTP mechanism called *content negotiation*. HTTP clients send HTTP headers with each request to indicate what kinds of representation they prefer. Servers can inspect those headers and select an appropriate response. If the headers indicate that the client prefers HTML, then the server can generate an HTML representation. If the client prefers RDF, then the server can generate RDF.

Content negotiation for non-information resources is usually implemented in the following way. When a URI identifying a non-information resource is dereferenced, the server sends a 303 redirect to an information resource appropriate for the client. Therefore, a data source often serves three URIs related to each non-information resource, for instance:

- http://www4.wiwiss.fu-berlin.de/factbook/resource/Russia (URI identifying the non-information resource Russia)
- http://www4.wiwiss.fu-berlin.de/factbook/data/Russia (information resource with an RDF/XML representation describing Russia)
- http://www4.wiwiss.fu-berlin.de/factbook/page/Russia (information resource with an HTML representation describing Russia)

The picture below shows how dereferencing a HTTP URI identifying a non-information resource plays together with content negotiation:

1. The client performs an HTTP GET request on a URI identifying a non-information resource. In our case a `vocabulary URI`. If the client is a Linked Data browser and would prefer an RDF/XML representation of the resource, it sends an `Accept: application/rdf+xml` header along with the request. HTML browsers would send an `Accept: text/html` header instead.
2. The server recognizes the URI to identify a non-information resource. As the server can not return a representation of this resource, it answers using the HTTP `303 See Other` response code and sends the client the URI of an information resource describing the non-information resource. In the RDF case: `RDF content location`.
3. The client now asks the server to GET a representation of this information resource, requesting again `application/rdf+xml`.
4. The server sends the client a RDF/XML document containing a description of the original resource `vocabulary URI`.



A complete example of a HTTP session for dereferencing a URI identifying a non-information resource is given in Appendix A.

URI Aliases

In an open environment like the Web it often happens that different information providers talk about the same non-information resource, for instance a geographic location or a famous person. As they do not know about each other, they introduce different URIs for identifying the same real-world object. For instance: DBpedia a data source providing information that has been extracted from Wikipedia uses the URI http://dbpedia.org/resource/Berlin to identify Berlin. Geonames is a data source providing information about millions of geographic locations uses the URI http://sws.geonames.org/2950159/ to identify Berlin. As both URIs refer to the same non-information resource, they are called URI aliases. URI aliases are common on the Web of Data, as it can not realistically be expected that all information providers agree on the same URIs to identify a non-information resources. URI aliases provide an important social function to the Web of Data as they are dereferenced to different descriptions of the same non-information resource and thus allow different views and opinions to be expressed. In order to still be able to track that different information providers speak about the same non-information resource, it is common practice that information providers set owl:sameAs links to URI aliases they know about. This practice is explained in Section 6 in more detail.

Associated Descriptions

Within this tutorial we use a new term which is not part of the standard Web Architecture terminology but useful in the Linked Data context. The term is *associated description* and it refers to the description of a non-information resource that a client obtains by dereferencing a specific URI identifying this non-information resource. For example: Deferencing the URI

http://dbpedia.org/resource/Berlin asking for `application/rdf+xml` gives you, after a redirect, an associated description that is equal to the RDF description of http://dbpedia.org/resource/Berlin within the information resource http://dbpedia.org/data/Berlin. Using this new term makes sense in a Linked Data context as it is common practice to use multiple URI aliases to refer to the same non-information resource and also because different URI aliases dereference to different descriptions of the resource.

2.2. The RDF Data Model

When publishing Linked Data on the Web, we represent information about resources using the Resource Description Framework (RDF). RDF provides a data model that is extremely simple on the one hand but strictly tailored towards Web architecture on the other hand.

In RDF, a description of a resource is represented as a number of *triples*. The three parts of each triple are called its *subject*, *predicate*, and *object*. A triple mirrors the basic structure of a simple sentence, such as this one:

```
    Chris     has the email address    chris@bizer.de .
(subject)        (predicate)              (object)
```

The subject of a triple is the URI identifying the described resource. The object can either be a simple *literal value*, like a string, number, or date; or the URI of another resource that is somehow related to the subject. The predicate, in the middle, indicates what kind of relation exists between subject and object, e.g. this is the name or date of birth (in the case of a literal), or the employer or someone the person knows (in the case of another resource). The predicate is a URI too. These predicate URIs come from *vocabularies*, collections of URIs that can be used to represent information about a certain domain. Please refer to Section 4 for more information about which vocabularies to use in a Linked Data context.

Some people like to imagine a set of RDF triples as an RDF graph. The URIs occurring as subject and object URIs are the nodes in the graph, and each triple is a directed arc (arrow) that connects the subject to the object.

Two principal types of RDF triples can be distinguished, Literal Triples and RDF Links:

Literal Triples

have an RDF literal such as a string, number, or date as the object. Literal triples are used to

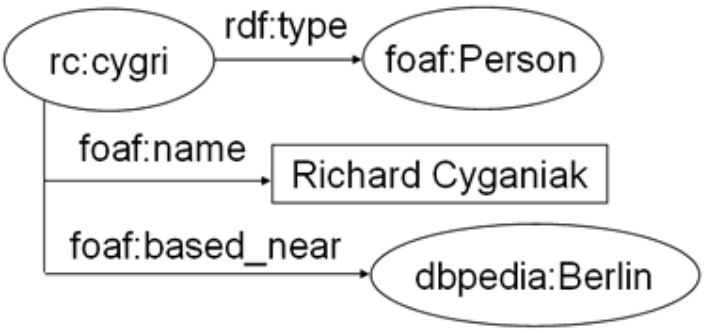
describe the properties of resources. For instance, literal triples are used to describe the name or date of birth of a person.

RDF Links

represent typed links between two resources. RDF links consist of three URI references. The URIs in the subject and the object position of the link identify the interlinked resources. The URI in the predicate position defines the type of the link. For instance, an RDF link can state that a person is *employed* by an organization. Another RDF link can state that the persons *knows* certain other people.

RDF links are the foundation for the Web of Data. Dereferencing the URI that appears as the destination of a link yields a description of the linked resource. This description will usually contain additional RDF links which point to other URIs that in turn can also be dereferenced, and so on. This is how individual resource descriptions are woven into the Web of Data. This is also how the Web of Data can be navigated using a Linked Data browser or crawled by the robot of a search engine.

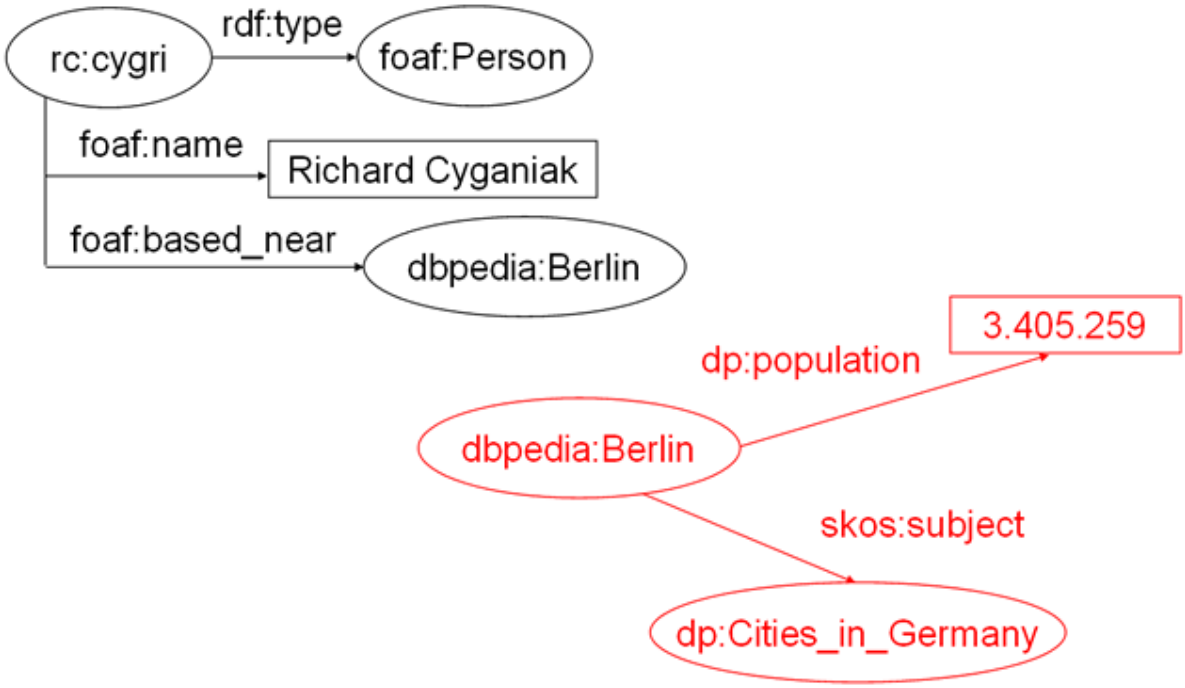
Let's take an RDF browser like Disco or Tabulator as an example. The surfer uses the browser to display information about Richard from his FOAF profile. Richard has identified himself with the URI http://richard.cyganiak.de/foaf.rdf#cygri. When the surfer types this URI into the navigation bar, the browser dereferences this URI over the Web, asking for content type `application/rdf+xml` and displays the retrieved information (click here to have Disco do this). In his profile, Richard has stated that he is based near Berlin, using the DBpedia URI http://dbpedia.org/resource/Berlin as URI alias for the non-information resource Berlin. As the surfer is interested in Berlin, he instructs the browser to dereference this URI by clicking on it. The browser now dereferences this URI asking for `application/rdf+xml`.



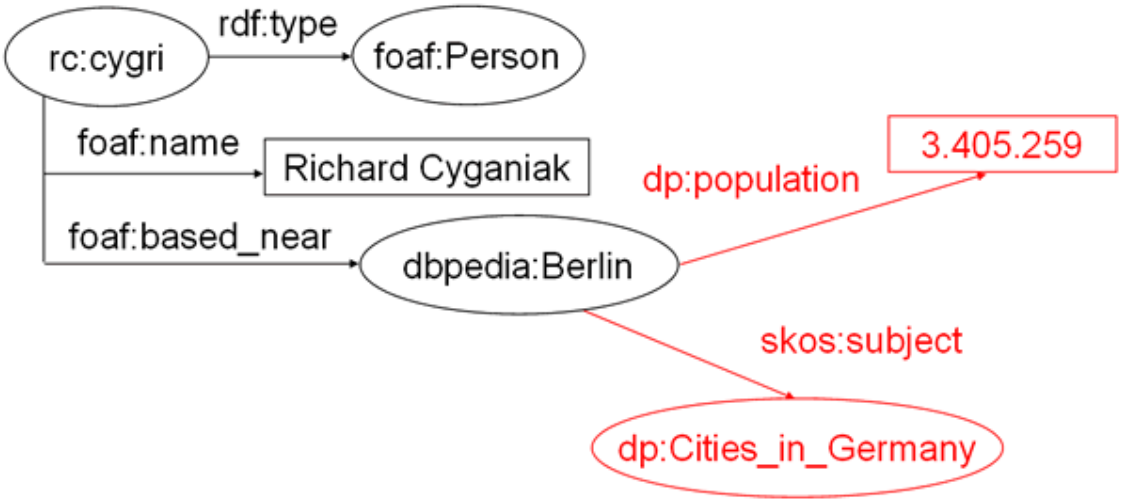

```
GET /resource/Berlin HTTP/1.0
Accept: application/rdf+xml
```

After being redirected with a HTTP 303 response code, the browser retrieves an RDF graph

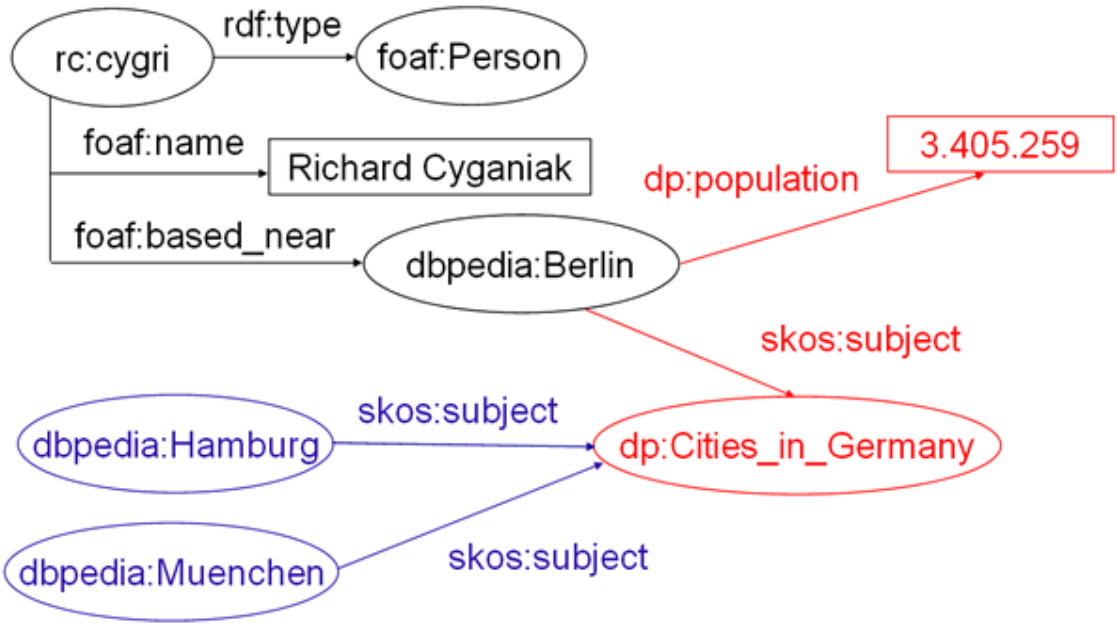
describing Berlin in more detail. A part of this graph is shown below. The graph contains a literal triple stating that Berlin has 3.405.259 inhabitants and another RDF link to a resource representing a list of German cities.



As both RDF graphs share the URI http://dbpedia.org/resource/Berlin, they naturally merge together, as shown below.



The surfer might also be interested in other German cities. Therefore he lets the browser dereference the URI identifying the list. The retrieved RDF graph contains more RDF links to German cities, for instance, Hamburg and München as shown below.



Seen from a Web perspective, the most valuable RDF links are those that connect a resource to external data published by other data sources, because they link up different islands of data into a Web. Technically, such an external RDF link is a RDF triple which has a subject URI from one data source and an object URI from another data source. The box below contains various external RDF links taken from different data sources on the Web.

Examples of External RDF Links

```
# Two RDF links taken from DBpedia

<http://dbpedia.org/resource/Berlin>

    owl:sameAs <http://sws.geonames.org/2950159/> .

<http://dbpedia.org/resource/Tim_Berners-Lee>

    owl:sameAs <http://www4.wiwiss.fu-berlin.de/dblp/resource/person/100007> .


# RDF links taken from Tim Berners-Lee's FOAF profile

<http://www.w3.org/People/Berners-Lee/card#i>

    owl:sameAs <http://dbpedia.org/resource/Tim_Berners-Lee> ;

    foaf:knows <http://www.w3.org/People/Connolly/#me> .


# RDF links taken from Richard Cyganiaks's FOAF profile
```

```
<http://richard.cyganiak.de/foaf.rdf#cygri>

    foaf:knows <http://www.w3.org/People/Berners-Lee/card#i> ;

    foaf:topic_interest <http://dbpedia.org/resource/Semantic_Web> .
```

Benefits of using the RDF Data Model in the Linked Data Context

The main benefits of using the RDF data model in a Linked Data context are that:

- Clients can look up every URI in an RDF graph over the Web to retrieve additional information.
- Information from different sources merges naturally.
- The data model enables you to set RDF links between data from different sources.
- The data model allows you to represent information that is expressed using different schemata in a single model.
- Combined with schema languages such as RDF-S or OWL, the data model allows you to use as much or as little structure as you need, meaning that you can represent tightly structured data as well as semi-structured data.

RDF Features Best Avoided in the Linked Data Context

In order to make it easier for clients to merge and query your data, we recommend not to use the full expressivity of the RDF data model, but a subset of the RDF features. Especially:

- We discourage the use of blank nodes. It is impossible to set external RDF links to a blank node, and merging data from different sources becomes much more difficult when blank nodes are used. Therefore, all resources of any importance should be named using URI references. Note that the current FOAF specification has also dropped blank nodes in favour of URI references (see rdf:about="#me" in their example, and Tim Berners-Lee's Give yourself a URI post on the topic).
- We discourage the use of RDF reification as the semantics of reification are unclear and as reified statements are rather cumbersome to query with the SPARQL query language. Metadata can be attached to the information resource instead, as explained in Section 5.
- You should think twice before using RDF collections or RDF containers as they do not work well together with SPARQL. Does your application really need a collection or a container or can the information also be expressed using multiple triples having the same predicate? The second option makes SPARQL queries straight forward.

3. Choosing URIs

Resources are named with URI references. When publishing Linked Data, you should devote some effort to choosing good URIs for your resources.

On the one hand, they should be *good names* that other publishers can use confidently to link to your resources in their own data. On the other hand, you will have to put technical infrastructure in place to make them *dereferenceable*, and this may put some constraints on what you can do.

This section lists, in loose order, some things to keep in mind.

- Use HTTP URIs for everything. The `http://` scheme is the only URI scheme that is widely supported in today's tools and infrastructure. All other schemes require extra effort for resolver web services, dealing with identifier registrars, and so on. The arguments in favour of using HTTP are discussed in several places, e.g. in Names and addresses by Norman Walsh, and URNs, Namespaces and Registries (draft) by the W3C TAG.
- Define your URIs in an HTTP namespace under your control, where you actually can make them dereferenceable. Do not define them in someone else's namespace.
- Keep implementation cruft out of your URIs. Short, mnemonic names are better. Consider these two examples:
  - http://dbpedia.org/resource/Berlin
  - http://www4.wiwiss.fu-berlin.de:2020/demos/dbpedia/cgi-bin/resources.php?id=Berlin
- Try to keep your URIs stable and persistent. Changing your URIs later will break any already-established links, so it is advisable to devote some extra thought to them at an early stage.
- The URIs you can choose are constrained by your technical environment. If your server is called demo.serverpool.wiwiss.example.org and getting another domain name is not an option, then your URIs will have to begin with http://demo.serverpool.wiwiss.example.org/. If you cannot run your server on port 80, then your URIs may have to begin with http://demo.serverpool.example.org:2020/. If possible you should clean up those URIs by adding some URI rewriting rules to the configuration of your webserver.
- We often end up with three URIs related to a single non-information resource:
  1. an identifier for the resource,
  2. an identifier for a related information resource suitable to HTML browsers (with a web page representation),
  3. an identifier for a related information resource suitable to RDF browsers (with an RDF/XML representation).

  Here are several ideas for choosing these related URIs:
  1. http://dbpedia.org/resource/Berlin

2. http://dbpedia.org/page/Berlin
3. http://dbpedia.org/data/Berlin

Or:

1. http://id.dbpedia.org/Berlin
2. http://pages.dbpedia.org/Berlin
3. http://data.dbpedia.org/Berlin

Or:

1. http://dbpedia.org/Berlin
2. http://dbpedia.org/Berlin.html
3. http://dbpedia.org/Berlin.rdf

- You will often need to use some kind of primary key inside your URIs, to make sure that each one is unique. If you can, use a key that is meaningful inside your domain. For example, when dealing with books, making the ISBN number part of the URI is better than using the primary key of an internal database table. This also makes equivalence mining to derive RDF links easier.

Examples of cool URIs:

- http://dbpedia.org/resource/Boston
- http://www4.wiwiss.fu-berlin.de/bookmashup/books/006251587X

See also:

- Sauermann et al.: Cool URIs for the Semantic Web (tutorial on URI dereferencing and content-negotiation)
- Common HTTP Implementation Problems, sections 1 and 3
- Tim Berners-Lee: Cool URIs don't change

---

4. Which vocabularies should I use to represent information?

In order to make it as easy as possible for client applications to process your data, you should reuse terms from well-known vocabularies wherever possible. You should only define new terms yourself if you can not find required terms in existing vocabularies.

4.1 Reusing existing terms

A set of well-known vocabularies has evolved in the Semantic Web community. Please check whether your data can be represented using terms from these vocabularies before defining any new terms:

- Friend-of-a-Friend (FOAF), vocabulary for describing people.
- Dublin Core (DC) defines general metadata attributes. See also their new domains and ranges draft.
- Semantically-Interlinked Online Communities (SIOC), vocabulary for representing online communities.
- Description of a Project (DOAP), vocabulary for describing projects.
- Simple Knowledge Organization System (SKOS), vocabulary for representing taxonomies and loosely structured knowledge.
- Music Ontology provides terms for describing artists, albums and tracks.
- Review Vocabulary, vocabulary for representing reviews.
- Creative Commons (CC), vocabulary for describing license terms.

A more extensive list of well-known vocabularies is maintained by the W3C SWEO Linking Open Data community project in the ESW Wiki. A listing of the 100 most common RDF namespaces (August 2006) is provided by UMBC eBiquity Group.

It is common practice to mix terms from different vocabularies. We especially recommend the use of rdfs:label and foaf:depiction properties whenever possible as these terms are well-supported by client applications.

If you need URI references for geographic places, research areas, general topics, artists, books or CDs, you should consider using URIs from data sources within the W3C SWEO Linking Open Data community project, for instance Geonames, DBpedia, Musicbrainz, dbtune or the RDF Book Mashup. The two main benefits of using URIs from these data sources are:

1. The URIs are dereferenceable, meaning that a description of the concept can be retrieved from the Web. For instance, using the DBpedia URI http://dbpedia.org/page/Doom to identify the computer game Doom gives you an extensive description of the game including abstracts in 10 different languages and various classifications.
2. The URIs are already linked to URIs from other data sources. For instance, you can navigate from the DBpedia URI http://dbpedia.org/resource/Berlin to data about Berlin provided by Geonames and EuroStat. Therefore, by using concept URIs form these datasets, you interlink your data with a rich and fast-growing network of other data sources.

A more extensive list of datasets with dereferenceable URIs is maintained by the Linking Open Data

community project in the ESW Wiki.

Good examples of how terms from different well-known vocabularies are mixed in one document and how existing concept URIs are reused are given by the FOAF profiles of Tim Berners-Lee and Ivan Herman.

4.2 How to define terms?

When you cannot find good existing vocabularies that cover all the classes and properties you need, then you have to define your own terms. Defining new terms is not hard. RDF classes and properties are resources themselves, identified by URIs, and published on the Web, so everything we said about publishing Linked Data applies to them as well.

You can define vocabularies using the RDF Vocabulary Description Language 1.0: RDF Schema or the Web Ontology Language (OWL). For introductions to RDFS, see the section on Vocabulary Documentation in the SWAP Tutorial, and the very detailed RDF Schema section of the RDF Primer. OWL is introduced in the OWL Overview.

Here we give some guidelines for those who are familiar with these languages:

1. **Do not define new vocabularies from scratch,** but complement existing vocabularies with additional terms (in your own namespace) to represent your data as required.
2. **Provide for both humans and machines.** At this stage in the development of the Web of Data, more people will be coming across your code than machines, even though the Web of Data is meant for machines in the first instance. Don't forget to add prose, e.g. rdfs:comments for each term invented. Always provide a label for each term using the rdfs:label property.
3. **Make term URIs dereferenceable.** It is essential that term URIs are dereferenceable so that clients can look up the definition of a term. Therefore you should make term URIs dereferenceable following the W3C Best Practice Recipes for Publishing RDF Vocabularies.
4. **Make use of other people's terms**. Using other people's terms, or providing mappings to them, helps to promote the level of data interchange on the Web of Data, in the same way that hypertext links built the traditional document Web. Common properties for providing such mappings are rdfs:subClassOf or rdfs:subPropertyOf.
5. **State all important information explicitly.** For example, state all ranges and domains explicitly. Remember: humans can often do guesswork, but machines can't. Don't leave important information out!
6. **Do not create over-constrained, brittle models; leave some flexibility for growth.** For instance, if you use full-featured OWL to define your vocabulary, you might state things that

lead to unintended consequences and inconsistencies when somebody else references your term in a different vocabulary definition. Therefore, unless you know exactly what you are doing, use RDF-Schema to define vocabularies.

The following example contains a definition of a class and a property following the rules above. The example uses the Turtle syntax. Namespace declarations are omitted.

```
# Definition of the class "Lover"

<http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LoveVocabulary#Lover>

    rdf:type rdfs:Class ;

    rdfs:label "Lover"@en ;

    rdfs:label "Liebender"@de ;

    rdfs:comment "A person who loves somebody."@en ;

    rdfs:comment "Eine Person die Jemanden liebt."@de ;

    rdfs:subClassOf foaf:Person .


# Definition of the property "loves"

<http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LoveVocabublary#loves>

    rdf:type rdf:Property ;

    rdfs:label "loves"@en ;

    rdfs:label "liebt"@de ;

    rdfs:comment "Relation between a lover and a loved person."@en ;

    rdfs:comment "Beziehung zwischen einem Liebenden und einer geliebten Person."@de ;

    rdfs:subPropertyOf foaf:knows ;

    rdfs:domain <http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LoveVocabulary#Lover> ;

    rdfs:range foaf:Person .
```

Note that the terms are defined in a namespace that is controlled by Chris Bizer and that they are related to the FOAF vocabulary using rdfs:subPropertyOf and rdfs:subClassOf links.

5. What should I return as RDF description for a URI?

So, assuming we have expressed all our data in RDF triples: What triples should go into the RDF representation that is returned (after a 303 redirect) in response to dereferencing a URI identifying a non-information resource?

1. **The description:** The representation should include all triples from your dataset that have the resource's URI as the subject. This is the immediate description of the resource.
2. **Backlinks:** The representation should also include all triples from your dataset that have the resource's URI as the object. This is redundant, as these triples can already be retrieved from their subject URIs, but it allows browsers and crawlers to traverse links in either direction.
3. **Related descriptions**: You may include any additional information about related resources that may be of interest in typical usage scenarios. For example, you may want to send information about the author along with information about a book, because many clients interested in the book may also be interested in the author. A moderate approach is recommended, returning a megabyte of RDF will be considered excessive in most cases.
4. **Metadata:** The representation should contain any metadata you want to attach to your published data, such as a URI identifying the author and licensing information. These should be recorded as RDF descriptions of the *information resource* that describes a non-information resource; that is, the subject of the RDF triples should be the URI of the information resource. Attaching meta-information to that information resource, rather than attaching it to the described resource itself or to specific RDF statements about the resource (as with RDF reification) plays nicely together with using Named Graphs and the SPARQL query language in Linked Data client applications. In order to enable information consumers to use your data under clear legal terms, each RDF document should contain a license under which the content can be used. Please refer to Creative Commons or Talis for standard licenses).
5. **Syntax:** There are various ways to serialize RDF descriptions. Your data source should at least provide RDF descriptions as RDF/XML which is the only official syntax for RDF. As RDF/XML is not very human-readable, your data source could additionally provide Turtle descriptions when asked for MIME-type `application/x-turtle`. In situations where your think people might want to use your data together with XML technologies such as XSLT or XQuery, you might additionally also serve a TriX serialization, as TriX works better with these technologies than RDF/XML.

In the following, we give two examples of RDF descriptions following the rules above. The first example covers the case of an authoritative representation served by a URI owner. The second example covers the case of non-authoritative information served by somebody who is not the owner

of the described URI.

1. Authoritative Description

The following example shows parts of the Turtle representation of the information resource http://dbpedia.org/data/Alec_Empire. The resource describes the German musician Alec Empire. Using Web Architecture terminology, it is a authoritative description as it is served after a 303 redirect by the owner of the URI http://dbpedia.org/resource/Alec_Empire. Namespace declarations are omitted:

```
# Metadata and Licensing Information

<http://dbpedia.org/data/Alec_Empire>

    rdfs:label "RDF description of Alec Empire" ;

    rdf:type foaf:Document ;

    dc:publisher <http://dbpedia.org/resource/DBpedia> ;

    dc:date "2007-07-13"^^xsd:date ;

    dc:rights

        <http://en.wikipedia.org/wiki/WP:GFDL> .


# The description

<http://dbpedia.org/resource/Alec_Empire>

    foaf:name "Empire, Alec" ;

    rdf:type foaf:Person ;

    rdf:type <http://dbpedia.org/class/yago/musician> ;

    rdfs:comment

        "Alec Empire (born May 2, 1972) is a German musician who is ..."@en ;

    rdfs:comment

        "Alec Empire (eigentlich Alexander Wilke) ist ein deutscher Musiker. ..."@de ;

    dbpedia:genre <http://dbpedia.org/resource/Techno> ;

    dbpedia:associatedActs <http://dbpedia.org/resource/Atari_Teenage_Riot> ;
```

```
    foaf:page <http://en.wikipedia.org/wiki/Alec_Empire> ;

    foaf:page <http://dbpedia.org/page/Alec_Empire> ;

    rdfs:isDefinedBy <http://dbpedia.org/data/Alec_Empire> ;

    owl:sameAs <http://zitgist.com/music/artist/d71ba53b-23b0-4870-a429-cce6f345763b> .


# Backlinks

<http://dbpedia.org/resource/60_Second_Wipeout>

    dbpedia:producer <http://dbpedia.org/resource/Alec_Empire> .

<http://dbpedia.org/resource/Limited_Editions_1990-1994>

    dbpedia:artist <http://dbpedia.org/resource/Alec_Empire> .
```

Note that the description contains an owl:sameAs Link stating that http://dbpedia.org/resource/Alec_Empire and http://zitgist.com/music/artist/d71ba53b-23b0-4870-a429-cce6f345763b are URI aliases referring to the same non-information resource.

In order to make it easier for Linked Data clients to understand the relation between http://dbpedia.org/resource/Alec_Empire, http://dbpedia.org/data/Alec_Empire, and http://dbpedia.org/page/Alec_Empire, the URIs can be interlinked using the rdfs:isDefinedBy and the foaf:page property as recommended in the Cool URI paper.

2. Non-Authoritative Description

The following example shows the Turtle representation of the information resource http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/ChrisAboutRichard which is published by Chris to provide information about Richard. Note that in Turtle, the syntactic shortcut <> can be used to refer to the URI of the current document. Richard owns the URI http://richard.cyganiak.de/foaf.rdf#cygri and is therefore the only person who can provide an authoritative description for this URI. Thus using Web Architecture terminology, Chris is providing non-authoritative information about Richard.

```
# Metadata and Licensing Information

<>

    rdf:type foaf:Document ;
```

```
        dc:author <http://www.bizer.de#chris> ;

        dc:date "2007-07-13"^^xsd:date ;

        cc:license <http://web.resource.org/cc/PublicDomain> .


    # The description

    <http://richard.cyganiak.de/foaf.rdf#cygri>

        foaf:name "Richard Cyganiak" ;

        foaf:topic_interest <http://dbpedia.org/resource/Category:Databases> ;

        foaf:topic_interest <http://dbpedia.org/resource/MacBook_Pro> ;

        rdfs:isDefinedBy <http://richard.cyganiak.de/foaf.rdf> ;

        rdf:seeAlso <> .


    # Backlinks

    <http://www.bizer.de#chris>

        foaf:knows <http://richard.cyganiak.de/foaf.rdf#cygri> .

    <http://www4.wiwiss.fu-berlin.de/is-group/resource/projects/Project3>

        doap:developer <http://richard.cyganiak.de/foaf.rdf#cygri> .
```

---

6. How to set RDF Links to other Data Sources

RDF links enable Linked Data browsers and crawlers to navigate between data sources and to discover additional data.

The application domain will determine which RDF properties are used as predicates. For instance, commonly used linking properties in the domain of describing people are foaf:knows, foaf:based_near and foaf:topic_interest . Examples of combining these properties with property values from DBpedia, the DBLP bibliography and the RDF Book Mashup are found in Tim Berners-Lee's and Ivan Herman's FOAF profiles.

It is common practice to use the owl:sameAs property for stating that another data source also provides information about a specific non-information resource. An owl:sameAs link indicates that two URI references actually refer to the same thing. Therefore, owl:sameAs is used to map between different URI aliases (see Section 2.1). Examples of using owl:sameAs to indicate that two URIs talk about the same thing are again Tim's FOAF profile which states that http://www.w3.org/People/Berners-Lee/card#i identifies the same resource as http://www4.wiwiss.fu-berlin.de/bookmashup/persons/Tim+Berners-Lee and http://www4.wiwiss.fu-berlin.de/dblp/resource/person/100007. Other usage examples are found in DBpedia and the Berlin DBLP server.

RDF links can be set manually, which is usually the case for FOAF profiles, or they can be generated by automated linking algorithms. This approach is usually taken to interlink large datasets.

6.1 Setting RDF Links Manually

Before you can set RDF links manually, you need to know something about the datasets you want to link to. In order to get an overview of different datasets that can be used as linking targets please refer to the Linking Open Data Dataset list. Once you have identified particular datasets as suitable linking targets, you can manually search in these for the URI references you want to link to. If a data source doesn't provide a search interface, for instance a SPARQL endpoint or a HTML Web form, you can use Linked Data browsers like Tabulator or Disco to explore the dataset and find the right URIs.

You can also use services such as Uriqr or Sindice to search for existing URIs and to choose the most popular one if you find several candidates. Uriqr allows you to find URIs for people you know, simply by searching for their name. Results are ranked according to how heavily a particular URI is referenced in RDF documents on the Web, but you will need to apply a little bit of human intelligence in picking the most appropriate URI to use. Sindice indexes the Semantic Web and can tell you which sources mention a certain URI. Therefore the service can help you to choose the most popular URI for a concept.

Remember that data sources might use HTTP-303 redirects to redirect clients from URIs identifying non-information resources to URIs identifying information resources that describe the non-information resources. In this case, make sure that you link to the URI reference identifying the non-information resource, and not the document about it.

6.2 Auto-generating RDF Links

The approach described above does not scale to large datasets, for instance interlinking 70,000 places in DBpedia to their corresponding entries in Geonames. In such cases it makes sense to use an automated record linkage algorithm to generate RDF links between data sources.

Record Linkage is a well-known problem in the databases community. The Linking Open Data Project collects material related to using record linkage algorithms in the Linked Data context on the Equivalence Mining wiki page.

There is still a lack of good, easy-to-use tools to auto-generate RDF links. Therefore it is common practice to implement dataset-specific record linkage algorithms to generate RDF links between data sources. In the following we describe two classes of such algorithms:

Pattern-based Algorithms

In various domains, there are generally accepted naming schemata. For instance, in the publication domain there are ISBN numbers, in the financial domain there are ISIN identifiers. If these identifiers are used as part of HTTP URIs identifying particular resources, it is possible to use simple pattern-based algorithms to generate RDF links between these resources.

An example of a data source using ISBN numbers as part of its URIs is the RDF Book Mashup, which for instance uses the URI http://www4.wiwiss.fu-berlin.de/bookmashup/books/0747581088 to identify the book 'Harry Potter and the Half-blood Prince'. Having the ISBN number in these URIs made it easy for DBpedia to generate owl:sameAs links between books within DBpedia and the Book Mashup. DBpedia uses the following pattern-based algorithm:

1. Iterate over all books in DBpedia that have an ISBN number.
2. Create a owl:sameAs link between the URI of a book in DBpedia and the corresponding Book Mashup URI using the following pattern for Book Mashup URIs: http://www4.wiwiss.fu-berlin.de/bookmashup/books/{ISBN number}.

Running this algorithm against all books in DBpedia resulted in 9000 RDF links which were merged with the DBpedia dataset. For instance, the resulting link for the Harry Potter book is:

```
<http://dbpedia.org/resource/Harry_Potter_and_the_Half-Blood_Prince>

    owl:sameAs <http://www4.wiwiss.fu-berlin.de/bookmashup/books/0747581088>
```

More complex property-based Algorithms

In cases where no common identifiers across datasets exist, it is necessary to employ more complex property-based linkage algorithms. We outline two algorithms below:

1. **Interlinking DBpedia and Geonames.** Information about geographic places appear in the Geonames database as well as in DBpedia. In order to identify places that appear in both datasets, the Geonames team uses a property-based heuristic that is based on article title together with semantic information like latitude and longitude, but also country, administrative division, feature type, population and categories. Running this heuristic against both data sources resulted in 70500 correspondences which were merged as Geonames owl:sameAs links with the DBpedia dataset as well as with the Geonames dataset.
2. **Interlinking Jamendo and MusicBrainz**. Please refer to Yves Raimond's blog post about his approach to interlinking Jamendo and MusicBrainz.

---

7. Recipes for Serving Information as Linked Data

This chapter provides practical recipes for publishing different types of information as Linked Data on the Web. Information has to fulfill the following minimal requirements to be considered "published as Linked Data on the Web":

- Things must be identified with dereferenceable HTTP URIs.
- If such a URI is dereferenced asking for the MIME-type `application/rdf+xml`, a data source must return an RDF/XML description of the identified resource.
- URIs that identify non-information resources must be set up in one of these ways: Either the data source must return an HTTP response containing an *HTTP 303 redirect* to an information resource describing the non-information resource, as discussed earlier in this document. Or the URI for the non-information resource must be formed by taking the URI of the related information resource and appending a *fragment identifier* (e.g. `#foo`), as discussed in Recipe 7.1.
- Besides RDF links to resources within the same data source, RDF descriptions should also contain RDF links to resources provided by other data sources, so that clients can navigate the Web of Data as a whole by following RDF links.

Which of the following recipes fits your needs depends on various factors, such as:

- **How much data do you want to serve?** If you only want to publish several hundred RDF

triples, you might want to serve them as a static RDF file using Recipe 7.1. If your dataset is larger, you might want to load it into a proper RDF store and put the Pubby Linked Data interface in front of it as described in Recipe 7.3.

- **How is your data currently stored?** If your information is stored in a relational database, you can use D2R Server as described in Recipe 7.2. If the information is available through an API, you might implement a wrapper around this API as described in Recipe 7.4. If your information is represented in some other format such as Microsoft Excel, CSV or BibTeX, you will have to convert it to RDF first as described in Recipe 7.3.
- **How often does your data change?** If your data changes frequently, you might prefer approaches which generate RDF views on your data, such as D2R Server (Recipe 7.2), or wrappers (Recipe 7.4).

After you have published your information as Linked Data, you should ensure that there are external RDF links pointing at URIs from your dataset, so that RDF browser and crawlers can find your data. There are two basic ways of doing this:

1. Add several RDF links to your FOAF profile that point at URIs identifying central resources within your dataset. Assuming that somebody else in the world knows you and references your FOAF profile, your new dataset is now reachable by following RDF links.
2. Convince the owners of related data sources to auto-generate RDF links to URIs from your dataset. Or to make it easier for the owner of the other dataset, create the RDF links yourself and send them to her so that she just has to merge them with her dataset. A project that is extremely open to setting RDF links to other data sources is the DBpedia community project. Just announce your data source on the DBpedia mailing list or send a set of RDF links to the list.

7.1 Serving Static RDF Files

The simplest way to serve Linked Data is to produce static RDF files, and upload them to a web server. This approach is typically chosen in situations where

- the RDF files are created manually, e.g. when publishing personal FOAF files or RDF vocabularies or
- the RDF files are generated or exported by some piece of software that only outputs to files.

There are several issues to consider:

Configuring the server for correct MIME types

Older web servers are sometimes not yet configured to return the correct MIME type when serving RDF/XML files. Linked Data browsers may not recognize RDF data served in this way because the server claims that it is not RDF/XML but plain text. To find out if your server needs fixing, use cURL tool and the steps outlined in this tutorial.

How to fix this depends on the web server. In the case of Apache, add this line to the `httpd.conf` configuration file, or to an `.htaccess` file in the web server's directory where the RDF files are placed:

```
AddType application/rdf+xml .rdf
```

This tells Apache to serve files with an `.rdf` extension using the correct MIME type for RDF/XML, `application/rdf+xml`. Note this means you have to name your files with the `.rdf` extension.

While you're at it, you can also add these lines to make your web server ready for other RDF syntaxes (N3 and Turtle):

```
AddType text/rdf+n3;charset=utf-8 .n3

AddType application/x-turtle .ttl
```

File size

On the document Web, it's considered bad form to publish huge HTML pages, because they load very slowly in browsers and consume unnecessary bandwidth. The same is true when publishing Linked Data: Your RDF files shouldn't be larger than, say, a few hundred kilobytes. If your files are larger and describe multiple resources, you should break them up into several RDF files, or use Pubby as described in recipe 7.3 to serve them in chunks.

When you serve multiple RDF files, make sure they are linked to each other through RDF triples that involve resources described in different files.

Choosing URIs for non-information resources

The static file approach doesn't support the 303 redirects required for the URIs of non-information resources. Fortunately there is another standards-compliant method of naming non-information resources, which works very well with static RDF files, but has a downside we will discuss later. This method relies on *hash URIs*.

When you serve a static RDF file at, say, `http://example.com/people.rdf`, then you should name the non-information resources described in the file by appending a *fragment identifier* to the file's URI. The identifier must be unique within the file. That way, you end up with URIs like this for your non-information resources:

- `http://example.com/people.rdf#alice`
- `http://example.com/people.rdf#bob`
- `http://example.com/people.rdf#charlie`

This works because HTTP clients dereference hash URIs by stripping off the part after the hash and dereferencing the resulting URI. A Linked Data browser will then look into the response (the RDF file in this case), and find triples that tell it more about the non-information resource, achieving an effect quite similar to the 303 redirect.

The downside of this naming approach is that the URIs are not very "cool" according to the criteria set out in section 3. There's a reference to a specific representation format in the identifiers (the `.rdf` extension). And if you choose to rename the RDF file later on, or decide to split your data into several files, then all identifiers will change and existing links to them will break.

That's why you should use this approach only if the overall structure and size of the dataset are unlikely to change much in the future, or as a quick-and-dirty solution for transient data where link stability isn't so important.

Extending the recipe for 303 redirects and content negotiation

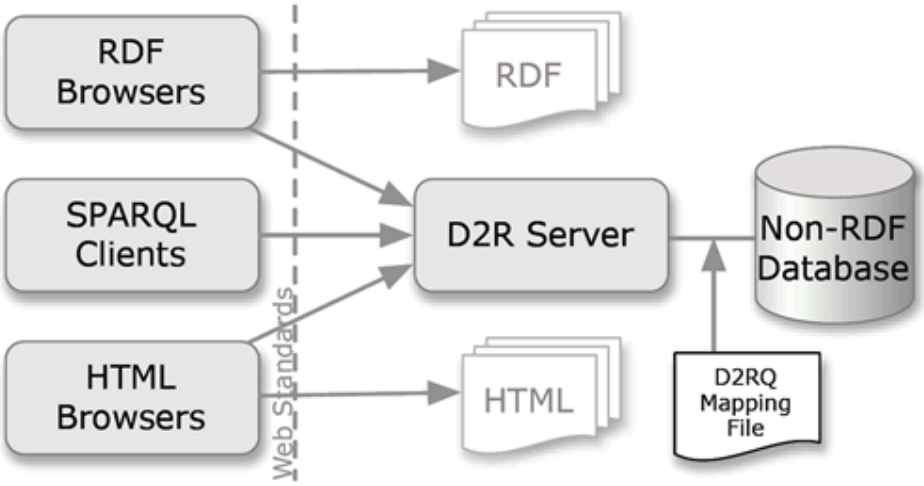
This approach can be extended to use 303 redirects and even to support content negotiation, if you are willing to go through some extra hoops. Unfortunately this process is dependent on your web server and its configuration. The W3C has published several recipes that show how to do this for the Apache web server: Best Practice Recipes for Publishing RDF Vocabularies. The document is officially targeted at publishers of RDF vocabularies, but the recipes work for other kinds of RDF data served from static files. Note that at the time of writing there is still an issue with content negotiation in this document which might be solved by moving from Apache mod_rewrite to mod_negotiation.

7.2 Serving Relational Databases

If your data is stored in a relational database it is usually a good idea to leave it there and just

publish a Linked Data view on your existing database.

A tool for serving Linked Data views on relational databases is D2R Server. D2R server relies on a declarative mapping between the schemata of the database and the target RDF terms. Based on this mapping, D2R Server serves a Linked Data view on your database and provides a SPARQL endpoint for the database.



There are several D2R Servers online, for example Berlin DBLP Bibliography Server, Hannover DBLP Bibliography Server, Web-based Systems @ FU Berlin Group Server or the EuroStat Countries and Regions Server.

Publishing a relational database as Linked Data typically involves the following steps:

1. Download and install the server software as described in the Quick Start section of the D2R Server homepage.
2. Have D2R Server auto-generate a D2RQ mapping from the schema of your database (see Quick Start).
3. Customize the mapping by replacing auto-generated terms with terms from well-known and publicly accessible RDF vocabularies.
4. Add your new data source to the ESW Wiki datasets list in the category Linked Data and SPARQL endpoint list and set several RDF links from your FOAF profile to the URIs of central resources within your new data source so that crawlers can discover your data.

Alternatively, you can also use:

1. OpenLink Virtuoso to publish your relational database as Linked Data.
    - Virtuoso RDF Views – Getting Started Guide on how to map your relational database to RDF and
    - Deploying Linked Data on how to get URI dereferencing and content negotiation into

place.

2. Triplify, a small plugin for Web applications, which reveals the semantic structures encoded in relational databases by making database content available as RDF, JSON or Linked Data.

7.3 Serving other Types of Information

If your information is currently represented in formats such as CSV, Microsoft Excel, or BibTEX and you want to serve the information as Linked Data on the Web it is usually a good idea to do the following:

- Convert your data into RDF using an RDFizing tool. There are two locations where such tools are listed: ConverterToRdf maintained in the ESW Wiki, and RDFizers maintained by the SIMILE team.
- After conversion, store your data in a RDF repository. A list of RDF repositories is maintained in the ESW Wiki.
- Ideally the chosen RDF repository should come with a Linked Data interface which takes care of making your data Web accessible. As many RDF repositories have not implemented Linked Data interfaces yet, you can also choose a repository that provides a SPARQL endpoint and put Pubby as a Linked Data interface in front of your SPARQL endpoint.

The approach described above is taken by the DBpedia project, among others. The project uses PHP scripts to extract structured data from Wikipedia pages. This data is then converted to RDF and stored in a OpenLink Virtuoso repository which provides a SPARQL endpoint. In order to get a Linked Data view, Pubby is put in front of the SPARQL endpoint.

If your dataset is sufficiently small to fit completely into the web server's main memory, then you can do without the RDF repository, and instead use Pubby's `conf:loadRDF` option to load the RDF data from an RDF file directly into Pubby. This might be simpler, but unlike a real RDF repository, Pubby will keep everything in main memory and doesn't offer a SPARQL endpoint.

7.4 Implementing Wrappers around existing Applications or Web APIs

Large numbers of Web applications have started to make their data available on the Web through Web APIs. Examples of data sources providing such APIs include eBay, Amazon, Yahoo, Google and Google Base. An more comprehensive API list is found at Programmable Web. Different APIs provide diverse query and retrieval interfaces and return results using a number of different formats

such as XML, JSON or ATOM. This leads to three general limitations of Web APIs:

- their content can not be crawled by search engines
- Web APIs can not be accessed using generic data browsers
- Mashups are implemented against a fixed number of data sources and can not take advantage of new data sources that appear on the Web.

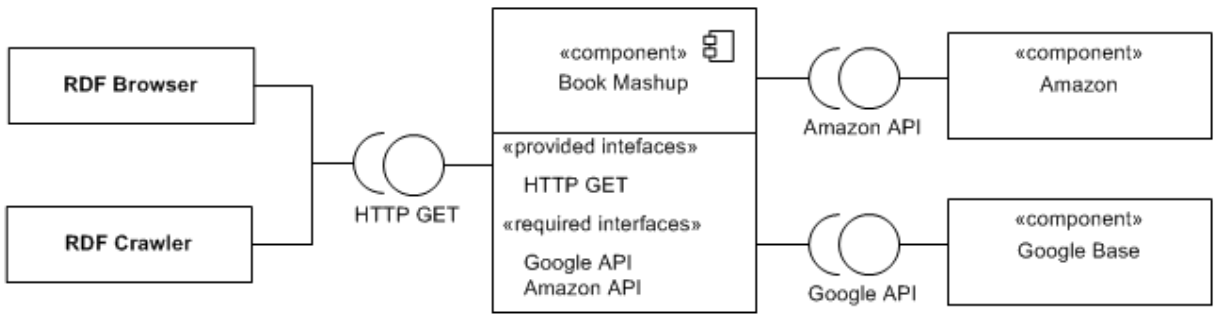
These limitations can be overcome by implementing Linked Data wrappers around APIs. In general, Linked Data wrappers do the following:

1. They assign HTTP URIs to the non-information resources about which the API provides data.
2. When one of these URIs is dereferenced asking for `application/rdf+xml`, the wrapper rewrites the client's request into a request against the underlying API.
3. The results of the API request are transformed to RDF and sent back to the client.

Examples of Linked Data Wrappers include:

The RDF Book Mashup

The RDF Book Mashup makes information about books, their authors, reviews, and online bookstores available as RDF on the Web. The RDF Book Mashup assigns a HTTP URI to each book that has an ISBN number. Whenever one of these URIs is dereferenced, the Book Mashup requests data about the book, its author as well as reviews and sales offers from the Amazon API and the Google Base API. This data is then transformed into RDF and returned to the client.



The RDF Book Mashup is implemented as a small PHP script which can be used as a template for implementing similar wrappers around other Web APIs. More information about the Book Mashup and the relationship of Web APIs to Linked Data in general is available in The RDF Book Mashup: From Web APIs to a Web of Data (Slides).

SIOC Exporters for WordPress, Drupal, phpBB

The SIOC project has developed Linked Data wrappers for several popular blogging

engines, content management systems and discussion forums. See SIOC Exporters for an up-to-date list of their wrappers. The project also provides a PHP Export API which enables developers to create further SIOC export tools without the need to get into technical details about how information is represented in RDF.

Virtuoso Sponger

Virtuoso Sponger is a framework for developing Linked Data wrappers (called cartridges) around different types of data sources. Data sources can range from HTML pages containing structured data to Web APIs. See Injecting Facebook Data into the Semantic Data Web for a demo on how Sponger is used to generate a Linked Data view on Facebook.

---

8. Testing and Debugging Linked Data

After you have published information as Linked Data on the Web, you should test whether your information can be accessed correctly.

An easy way of testing is to put several of your URIs into the Vapour Linked validation service, which generates a detailed report on how your URIs react to different HTTP requests.

Beside of this, it is also important to see whether your information displays correctly in different Linked Data browsers and whether the browsers can follow RDF links within your data. Therefore, take several URIs from your dataset and enter them into the navigation bar of the following Linked Data browsers:

- Tabulator. If Tabulator takes a long time before it displays your information, then this is an indicator that your RDF graphs are too big and should be split up. Tabulator also does some basic inferencing over Web data, without doing consistency checks. Therefore, if Tabulator behaves strangely, this might indicate issues with rdfs:subClassOf and rdfs:subPropertyOf declarations in the RDFS and OWL schemas used in your data.
- Marbles
- OpenLink RDF Browser
- Disco. The Disco browser uses a 2 second time-out when retrieving data from the Web. Therefore, it might be an indicator that your server is too slow, if Disco does not display your data correctly.

If you run into problems, you should do the following:

1. Test with cURL whether dereferencing your URIs leads to correct HTTP responses. Richard Cyganiak has published a tutorial on Debugging Semantic Web sites with cURL which leads you through the process.
2. Use the W3C's RDF Validation service to make sure that your service provides valid RDF/XML.

If you can not figure out yourself what is going wrong, ask on the Linking Open Data mailing list for help.

---

9. Discovering Linked Data on the Web

The standard way of discovering Linked Data on the Web is by **following RDF Links** within data the client already knows. In order to further ease discovery, information providers can decide to support additional discovery mechanisms:

Ping the Semantic Web

Ping the Semantic Web is a registry service for RDF documents on the Web, which is used by several other services and client applications. Therefore, you can improve the discoverability of your data by registering your URIs with Ping The Semantic Web.

HTML Link Auto-Discovery

It also makes sense in many cases to set links from existing webpages to RDF data, for instance from your personal home page to your FOAF profile. Such links can be set using the HTML `<link>` element in the `<head>` of your HTML page.

```
<link rel="alternate" type="application/rdf+xml" href="link_to_the_RDF_version" />
```

HTML `<link>` elements are used by browser extensions, like Piggybank and Semantic Radar, to discover RDF data on the Web.

Semantic Web Crawling: a Sitemap Extension

Semantic Web Crawling: a Sitemap Extension. The sitemap extension allows Data publishers can state where RDF is located and which alternative means are provided to access it (Linked Data, SPARQL endpoint, RDF dump). Semantic Web clients and Semantic Web crawlers can use this information to access RDF data in the most efficient way for the task they have to perform.

Dataset List on the ESW Wiki

In order to make it easy not only for machines but also for humans to discover your data, you should add your dataset to the Dataset List on the ESW Wiki. Please include some example URIs of interesting resources from your dataset, so that people have starting points for browsing.

---

10. Further Reading and Tools

For more information about Linked Data please refer to:

Overview Material and Theoretical Background

- Tim Berners-Lee: Linked Data (architecture note outlining the basic principles of Linked Data)
- Tim Berners-Lee: Giant Global Graph (blog post)
- Tim Berners-Lee: Browsable Data (slides about Linked Data)
- Tim Berners-Lee: Links on the Semantic Web (blog post stating the critical role of RDF links for the Semantic Web).
- LinkedData.org
- Wikipedia page on Linked Data
- ESW Wikipage on Linked Data
- W3C Semantic Web Activity Homepage
- W3C Semantic Web Frequently Asked Questions
- Paul Miller: Linked Data BOF and Linked Data once again (blog posts about the Linked Data sessions at WWW2007)
- Paul Miller: Opening the Silos - sustainable models for open data (video from the XTech 2007 - Open Data track)
- Michael K. Bergman: More Structure, More Terminology and (hopefully) More Clarity (contextualizes Linked Data in the overall development of the Web)
- Li Ding, Tim Finin: Characterizing the Semantic Web on the Web (ISWC2006 paper trying to measure the size of the Semantic Web, a bit outdated).
- Mike Bergman: Linked Data Comes of Age - LinkedData Planet and LDOW Set the Pace for 2008, February 2008.
- Uche Ogbuji: Real Web 2.0: Linking open data - Discover the community that sees Web 2.0

as a way to revolutionize information on the Web, IBM Developerworks, February 2008.

Technical Documentation

- Leo Sauermann et al.: Cool URIs for the Semantic Web (tutorial on URI dereferencing and content-negotiation)
- Alistair Miles et al.: Best Practice Recipes for Publishing RDF Vocabularies (W3C draft on serving RDF vocabularies according to the Linked Data principles)
- Richard Cyganiak: Debugging Semantic Web sites with cURL (tutorial on how to test Semantic Web sites)
- Semantic Web Tools (listing of RDF stores and development tools)
- Christian Bizer, Daniel Westphal: Developers Guide to Semantic Web Toolkits (another list of RDF development tools)
- Semantic Web Crawling: a Sitemap Extension. The sitemap extension allows Data publishers can state where RDF is located and which alternative means are provided to access it (Linked Data, SPARQL endpoint, RDF dump).

Projects and Practical Experience with Publishing Linked Data

- Linking Open Data Project (W3C SWEO community effort that publishes as RDF and interlinks open data sources such as Geonames, DBpedia, Musicbrainz, ... See also List of public Linked Data sources on the Web)
- Christian Bizer et al.: Interlinking Open Data on the Web (poster) (Overview about the W3C SWEO Linking Open Data project)
- Christian Bizer et al.: The RDF Book Mashup: From Web APIs to a Web of Data (Slides). (SFSW2007 paper)
- Marcus Cobden et al.: ECS URI System (Documentation on how the ECS department of the University of Southampton implements Linked Data on their department website.
- A similar example is given be the website of the Web-based Systems Group @ Freie Universität Berlin
- Christian Bizer et al: DBpedia - Querying Wikipedia like a Database (Slides). (WWW2007 Dev-Track presentation)

Linked Data Clients

- Tabulator RDF Browser (Linked Data browser implemented by Tim Berners-Lee et al.)
- Tim Berners-Lee et al.: Tabulator: Exploring and Analyzing Linked Data on the Semantic Web

(Slides) (SWUI2007 paper about the Tabulator Linked Data browser and its data retrieval algorithms)
- DISCO Hyperdata Browser (a simple Linked Data browser provided by Freie Universität Berlin)
- OpenLink Data Web Browser (Linked Data browser provided by OpenLink)
- Objectviewer (Linked Data browser provided by SemanticWebCentral)
- Semantic Web Client Library (Java framework for building Linked Data clients)
- Semantic Web client for SWI Prolog (This is a small Prolog program allowing to see the semantic-web as a single graph, and to browse it, similar to the Semantic Web Client Library)
- Tabulator AJAR RDF library for Javascript (retrieval engine underlying the Tabulator browser)
- OpenLink Ajax Toolkit (includes both a Data Access Layer of RDF, SQL, and XML called Ajax Database Connectivity and a collection of RDF aware controls covering: Graph Visualizers, TimeLines, Tag Clouds, Pivot Tables, and more)
- See also List of Linked Data Clients (maintained by the W3C Linking Open Data project)

Web of Data Search Engines

- Falcons developed by IWS China, currently indexes 7 million RDF documents.
- Sindice developed by DERI Ireland, currently indexes over 20 million RDF documents. See also their ISWC2007 paper Sindice.com: Weaving the Open Linked Data
- Watson developed by KMi, UK. See also the list of papers and presentations about Watson
- Semantic Web Search Engine (SWSE) developed by DERI Ireland. See also their paper MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data.
- Swoogle developed by ebiquity group at UMBC USA, currently indexes 2.3 million RDF documents. See also their papers about Swoogle .

Appendix A: Example HTTP Session

This is an example HTTP session where a Linked Data browser tries to dereference the URI http://dbpedia.org/resource/Berlin, a URI for the city of Berlin, published by the DBpedia project.

To obtain a representation, the client connects to the `dbpedia.org` server and issues an HTTP GET request:

```
GET /resource/Berlin HTTP/1.1
```

```
Host: dbpedia.org

Accept: text/html;q=0.5, application/rdf+xml
```

The client sends an `Accept:` header to indicate that it would take either HTML or RDF; the `q=0.5` quality value for HTML shows that it prefers RDF. The server could answer:

```
HTTP/1.1 303 See Other

Location: http://dbpedia.org/data/Berlin

Vary: Accept
```

This is a 303 redirect, which tells the client that the requested resource is a non-information resource, and its associated description can be found at the URI given in the `Location:` response header. Note that if the `Accept:` header had indicated a preference for HTML, we would have been redirected to another URI. This is indicated by the `Vary:` header, which is required so that caches work correctly. Next the client will try to dereference the URI of the associated description.

```
GET /data/Berlin HTTP/1.1

Host: dbpedia.org

Accept: text/html;q=0.5, application/rdf+xml
```

The server could answer:

```
HTTP/1.1 200 OK

Content-Type: application/rdf+xml;charset=utf-8


<?xml version="1.0"?>

<rdf:RDF

    xmlns:units="http://dbpedia.org/units/"

    xmlns:foaf="http://xmlns.com/foaf/0.1/"

    xmlns:geonames="http://www.geonames.org/ontology#"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
...
```

The 200 status code tells the client that the response contains the representation of an information resource. The `Content-Type:` header tells us that the representation is in RDF/XML format. The rest of the message contains the representation. Only the beginning is shown.

Appendix B: How to get yourself into the Web of Data

A great way to get started with publishing Linked Data on the Web is to serve a static RDF file; this can work well for small amounts of relatively simple data. One common example of this practice is providing a Friend-of-a-Friend (FOAF) file alongside (and interlinked with) your HTML home page. This Appendix provides step-by-step instructions on how to create and publish a FOAF description of yourself, and how to link it into the Web of Data.

FOAF-a-Matic is a web form that will generate a basic FOAF description for you in RDF/XML (which we will call your "FOAF file" from now onwards). This provides an excellent foundation to which you can add additional data. After generating your FOAF profile using FOAF-a-Matic, you'll need to save the generated RDF/XML code as a static file (using the filename `foaf.rdf` is a common convention) and decide where it will be hosted. Technically your FOAF file can be hosted anywhere on the Web, although it's common practice to use your own Web space and place it in the same directory as your home page. For example, Richard Cyganiak's FOAF file is located at http://richard.cyganiak.de/foaf.rdf; this is the URI of the RDF document, the document describes Richard, who is identified by the URI `http://richard.cyganiak.de/foaf.rdf#cygri`.

By default FOAF-a-Matic will use a fragment identifier (such as *#me*) to refer to you within your FOAF file. This fragment identifier is appended to the URI of your FOAF file to form *your* URI, of the form `http://yourdomain.com/foaf.rdf#me`. Congratulations, you now have a URI which can be used to identify you in other RDF statements on the Web of Data. Your URI is an example of a "hash URI", discussed in more detail in Section 7.1.

At this stage you will want to start linking your FOAF file into the Web. One good place to start is by linking to your FOAF file from your homepage, using the HTML LINK Auto-Discovery technique from Section 9, but don't stop there. To firmly embed your FOAF file into the Web of Data you need to read on and implement the guidance in the following sections.

Change Blank Nodes to URI references

At present FOAF-a-Matic uses "blank nodes" to identify people you know, not URI references, as the system has no way of knowing the appropriate URIs to associate with each person. Example output with blank nodes is shown below:

```
...
<foaf:knows>
  <foaf:Person>
    <foaf:mbox_sha1sum>362ce75324396f0aa2d3e5f1246f40bf3bb44401</foaf:mbox_sha1sum>
    <foaf:name>Dan Brickley</foaf:name>
    <rdfs:seeAlso rdf:resource="http://danbri.org/foaf.rdf"/>
  </foaf:Person>
</foaf:knows>
...
```

This is valid RDF but isn't good Linked Data, as blank nodes make it much harder to link and merge data across different sources (see Section 2.2 for more discussion of the issues with blank nodes). Therefore, the first thing to do in making your FOAF file into Linked Data is to look at the blank nodes representing people you know, and to replace them with URIs where possible.

Link to your friends

Section 6.1 explains how to set RDF links manually, and describes two services (*Uriqr* and *Sindice*) that you can use to try to find existing URIs for people you know. Following this approach we can find out that the existing URI http://danbri.org/foaf.rdf#danbri identifies Dan Brickley, and replace the blank node generated by FOAF-a-Matic with this URI reference. The result would look like this:

```
...
<foaf:knows>
  <foaf:Person rdf:about="http://danbri.org/foaf.rdf#danbri">
    <foaf:mbox_sha1sum>362ce75324396f0aa2d3e5f1246f40bf3bb44401</foaf:mbox_sha1sum>
    <foaf:name>Dan Brickley</foaf:name>
    <rdfs:seeAlso rdf:resource="http://danbri.org/foaf.rdf"/>
```

```
    </foaf:Person>
  </foaf:knows>
  ...
```

After setting links to people you know, there are many other ways in which you can link your FOAF file into the Web of Data. Here we will discuss two of these.

Tell people where you live

A common thing to include in your FOAF file is information about where you are based, using the `foaf:based_near property`. This isn't supported by FOAF-a-Matic, so you'll need to add the code in manually. Add the following line somewhere inside the `<foaf:Person rdf:ID="me"></foaf:Person>` element, replacing the object of the triple with the URI reference of the place nearest to where you are based.

```
<foaf:based_near rdf:resource="http://dbpedia.org/resource/Milton_Keynes"/>
```

Using URIs from DBpedia or Geonames ensures that you are linking your FOAF file to well-established URIs which are also likely to be used by others, therefore making the Web more interconnected.

How to link to your publications

If you have ever written a book or published an academic paper in the field of Computer Science, a URI may already have been created for you in the RDF version of DBLP or in the RDF Book Mashup. At a general level, how to handle this issue is touched upon under *URI Aliases* in Section 2.1. In summary, you simply need to link to them with a statement saying that they identify the same thing as the URI identifying you in your FOAF file. Section 6 describes how this should be done using the `owl:sameAs` property.

Appendix C: Changes

- 2008-07-17: Added link to Vapour Linked Data validation service.

- 2008-02-18: Fixed typos pointed out by Gus Gollings. Added new references to the "Further Reading and Tools" section.
- 2007-08-29: Added white papers on Virtuoso to section 7.2.
- 2007-07-27: Replaced the term 'data item' with 'associated description' after feeback from Frank Manola and extended discussions on the Semantic Web mailing list. Added reference to Virtuoso Sponger and to Michael K. Bergman blog post about data on the Web.
- 2007-07-18: Added new section 7.1 from Richard. Added Apendix B from Tom.
- 2007-07-17: Updated the document with feedback from Franois-Paul Servant and Leo Sauermann.
- 2007-07-14: Moved FOAF example into Appendix B. Updated images in section two.
- 2007-07-13: Updated the document with feedback from Ivan Herman, Joshua Shinavier, Giovanni Tummarello, Yves Raimond.
- 2007-07-12: Small edits across the document by Chris. Sindice added.
- 2007-07-11: Initial version of this document.