An Introduction to the Semantic Web, Considerations for building multilingual Semantic Web sites and applications

Jeremy J. Carroll
Digital Media Systems Laboratory
HP Laboratories Bristol
HPL-2005-67
April 22, 2005*

semantic web, language, RDF, OWL, internationalization

This paper gives a brief overview of the Semantic Web standards, the Resource Description Framework (RDF) and the Web Ontology Language (OWL). The emphasis is on the representation of natural language information within these standards, particularly: encoding of language identification, use of embedded XHTML, use of natural language labels for concepts in multiple languages, use of internationalized resource identifiers, and query using language information. Some problems are identified, such as: counting in OWL, locale and culture. The target reader knows little about the Semantic Web, but is well-versed in software internationalization and localization.

An Introduction to the Semantic Web, Considerations for building multilingual Semantic Web sites and applications

Jeremy J. Carroll, Hewlett-Packard Labs.

There is a lot of information on the Web, but … to understand it at all, you need to read and understand the text. A software system trying to use Web information has a hard time, because it cannot understand natural language. Even the structure of the information is encoded in HTML tags, which describe the presentation intended for a human, rather than following the underlying knowledge available on the Web site. This is particularly frustrating when the Web site is generated from a back-end database in which the structure of the information is explicit in database tables, which could be used by other software components.

The Semantic Web vision of Tim Berners-Lee, the creator of theWeb, is that Web based information should also be available in a machine processable form, allowing machines to do routine Web based tasks on behalf of human users, where currently the humans are needed to make sense of the Web pages.

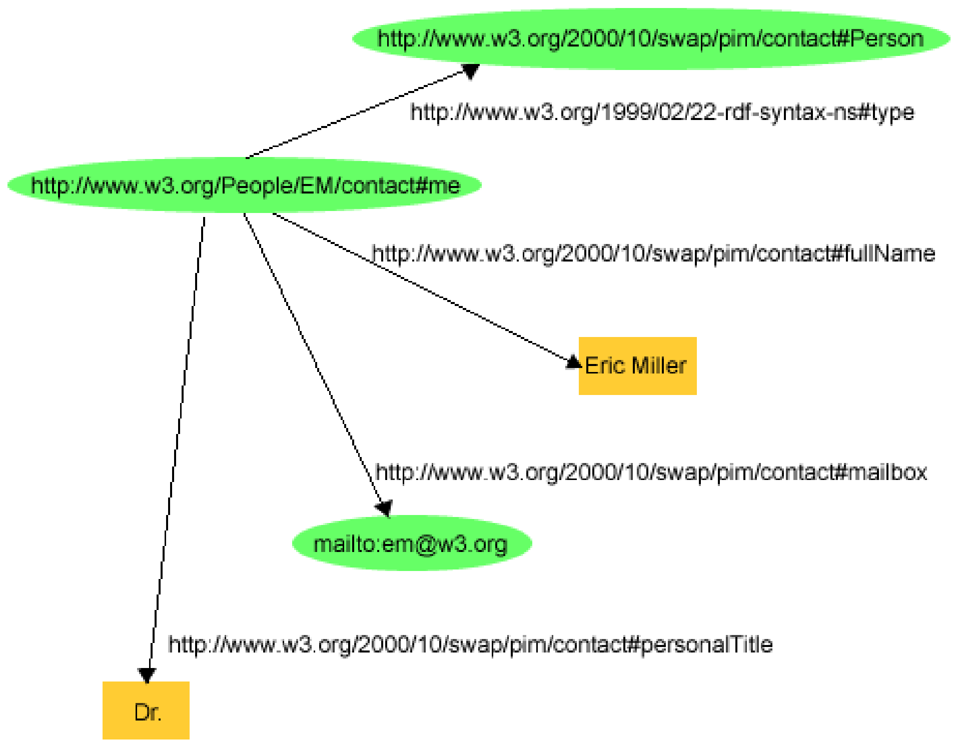
In this article, I give a brief introduction to the Semantic Web, and describe difficulties, and occasionally solutions, relating to building multilingual Semantic Web sites and applications.

The initial drivers for the Semantic Web came from *metadata* about Web pages. Who wrote it? When? Who owns the copyright? Etc. To convey such metadata there needs to be agreement about the key terms such as *author* and *date*. This agreement has been reached by the Dublin Core community, see http://dublincore.org/. As an example, they have an agreed definition for the term *creator*, generalizing *author* for use in metadata records.

However, the Semantic Web does not draw a sharp distinction between metadata about the page, and data contained within the page. In both cases, the idea is to provide sufficient structure around the data to turn it into information, and to connect the concepts used to express such information with concepts used by others, so that this information can become knowledge that can be acted upon.

The Resource Description Framework (RDF)

The central Semantic Web Recommendation from the World Wide Web Consortium (W3C) gives a universal information format for describing things, called the Resource Description Framework (RDF). Both the things described and the terminology used in describing them are identified using URIs. A full introduction is found in the RDF Primer, from the W3C. Here I give just the central ideas, starting with the first example from the RDF Primer:


http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

I will start by describing the second arrow which is labelled `http://www.w3.org/2000/10/swap/pim/contact#fullName.` This joins two nodes, the green oval, labelled `http://www.w3.org/People/EM/contact#me` and the yellow rectange labelled `“Eric Miller”`. The two nodes and the arrow are together called an *RDF triple*. The node at the beginning of the arrow, the green oval, is called the subject of the triple. The other node is the object. Each node represents a thing, and the arrow represents some relationship between the two things. These things are often referred to as *resources*, but I will stick to *thing* here. To understand what the triple represents, it is usually best to start with the label on the arrow, called the *property*. This is always a URI, and often as in this case, we can type it into a browser, effectively clicking on it. You may need to set your browser up to accept mimetype `application/rdf+xml`. Alternatively, you can look at the equivalent documents `http://www.w3.org/2000/10/swap/pim/contact.n3` or `http://www.w3.org/2000/10/swap/pim/contact.rdf`. Any of these gives a further RDF document, including, once you know how to read it, a description of the `#fullName` property as *“full name”*.

The yellow rectangles in the picture are *literal* values, in this case, the string `“Eric Miller”`, so the green oval represents something whose full name is “Eric Miller”. The diagram shows a collection of these triples, called an *RDF graph*. Each of the other arrows in the diagram gives a further property and value pair for the green oval, representing more information about the thing whose full name is “Eric Miller”. When we put it all together, we might hopefully think, that this thing is Dr Eric Miller, (who I work with in the W3C; his role is the Semantic Web activity lead).

We see that some triples have objects which are also green ovals. These represent further things, also identified by URIs. These things could be the subject of further

triples, that would help describe what they represent. Typically, some of the *nodes* (such as the green ovals) in a RDF graph will be both the object of some triples, and the subject of others. Loops of triples are also possible. The graph can be large and complicated, allowing encodings or descriptions of complex data structures.
RDF consists of using graphs like this to describe things.

Some property URIs are sufficiently widely used to be well-known: an example being `http://purl.org/dc/elements/1.1/creator` the creator property from Dublin Core. This URI is known by many different applications as representing the concept of creator.

Such RDF graphs are usually provided on the Web encoded in XML. The particular encoding used is called RDF/XML, which is ugly and seems unnecessarily difficult, and which I do not discuss further (although one or two examples will use it).

Typical RDF software includes a *triple store*, which is a database of RDF triples, with the ability to read and write RDF/XML documents, and to query the triple store. Typically a triple store is embedded within some programming language and the triples can be manipulated programmatically within that language. Later, there are a few Java examples, on top of the popular opensource Jena Semantic Web Framework from HP, which includes a triple store.

Often, a triple store, will be referred to as a *knowledge base*, indicating that the RDF graph is intended to be understood as a representation of some knowledge.

Classes

The topmost triple in the example, is labelled with the property URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`, which is often abbreviated to `rdf:type`. The subject is `http://www.w3.org/People/EM/contact#me` and the object is `http://www.w3.org/2000/10/swap/pim/contact#Person.` The object of an `rdf:type` triple is called a *class*, and represents a collection of things, in this case, it is a collection of people. We can click on the URI labelling a class, in this case, `http://www.w3.org/2000/10/swap/pim/contact#Person`, to find the definition of the class: *"A person in the normal sense of the word."* As before, you might need to modify the URI, depending on your browser setup, and look at `http://www.w3.org/2000/10/swap/pim/contact.n3` or `http://www.w3.org/2000/10/swap/pim/contact.rdf.` In this way, RDF provides the ability to represent classifications within the Semantic Web.

Classes fall naturally into hierarchies of subclasses and superclasses. Properties can be thought of as having subproperties and superproperties. The RDF Schema vocabulary is used to describe such hierarchies. As a simple example, `#Person`, is described as a subclass of `#SocialEntity` (which is the "*sort of thing that can have a phone number*").

The Web Ontology Language, OWL

Simple class hierarchies can be made more useful by relating classes and properties. For example, we may want to say that a man is a male person. This relates two classes: man and person, with a property, gender, and its value. The Web Ontology Language (OWL) provides a way of saying such things.

```
EquivalentClass(
   Class(my:Man)
   Intersection(
      Class(contact:Person)
      Restriction( my:gender , Value( “Male” ) ) ) )
```
(Note that `my:` and `contact:` are being used to abbreviate long URIs)

This expression can be read as “being a `my:Man` is the same as being both a `contact:Person` and having a `my:gender` property with value `“Male”`”
Such expressions are called *axioms*, and are usually translated into RDF triples, and then written as RDF/XML (which is too verbose to include here).

An *ontology*, is a collection of classes and properties, along with axioms, such as this one, relating them together.

An introduction to OWL can be found in the OWL Guide from the W3C. OWL is designed to work well with RDF. Ontologies written in OWL can be used to help organize and understand knowledge recorded in RDF.

Where do languages fit in?

So far, I seemed to have assumed that the whole world speaks US English. The vast majority of examples in the W3C documentation also seem to make this assumption. Fortunately, this is not quite true. There are five features included in RDF (and hence in OWL) which support multilingual (or explicitly monolingual) applications:
- Unicode
- language tagging, following RFC 3066 and `xml:lang`
- natural language labels for concepts
- embedded XML
- and Internationalized Resource Identifiers (IRIs)

Unicode: like most recent work from the W3C, Unicode is a given. In fact, neither RDF or OWL provide any support for any other character encodings. Typical Semantic Web developers’ tools use Unicode throughout. Legacy encodings can be used within RDF/XML documents, using the usual XML encoding declaration. Semantic Web software that does not support Unicode is broken, and should be avoided. In practice, however, many systems only allow 16 bits for each character, and hence are not Unicode 4 compliant. Depending on your needs, a detailed discussion with software suppliers may be useful (e.g. for Gothic, additional mathematical symbols for MathML, or the CJK compatibility ideographs), and is likely to require some persistence in helping them understand the issues.

Language Tagging: following XML, text in RDF can have a language tag, such as `“en-US”` or `“ja”` taken from RFC 3066. In RDF/XML, this is done using `xml:lang`, just like in other XML documents. The `xml:lang` declaration can be on the root element, or lower down, as appropriate. In an RDF graph there is *no* default language tag. Every string needs to be individually marked. In a triple store, the language tag is stored with each string. As an example the `“Dr.”` in the example should have been tagged with `en-US`, and would have conventionally been written as `"Dr."@en-US`. Note that RFC 3066 is currently being updated.

Natural Language Labels: Concepts, whether things, properties or classes, in the Semantic Web are represented using URIs. Normally, as in the example, these URIs have some intelligible meaning in some natural language, usually English. However, there is a property `rdfs:label`, that is used to provide natural language strings that should be used in preference to the URI. These string labels should be tagged with a language identifier, and Semantic Web software should display an appropriately chosen label. As an example, this class definition includes both an English and an Italian label (this is in RDF/XML):

```
<owl:Class rdf:ID="ShakespearePlay">
 <rdfs:label xml:lang="en">Shakespeare's plays</rdfs:label>
 <rdfs:label xml:lang="it">Opere di Shakespeare</rdfs:label>
</owl:Class>
```

In this example, the English and Italian labels are included in the same RDF/XML file. It may be easier to organise the labels for each language in separate files:

```
<owl:Class rdf:ID="ShakespearePlay">
 <rdfs:label xml:lang="en">Shakespeare's plays</rdfs:label>
</owl:Class>
```

in one, and

```
<owl:Class rdf:ID="ShakespearePlay">
 <rdfs:label xml:lang="it">Opere di Shakespeare</rdfs:label>
</owl:Class>
```

in the other. The ability to do this gives flexibility. Indeed it is possible to add labels to concepts defined by different organizations on different sites, for knowledge bases for which you do not have write-access. (There is no mechanism for persuading people to use your labels).

Embedded XML/XHTML: There are some cases where natural language cannot be adequately presented as a text string. These include mixed language examples, where some of the string is one language, and some another, and cases where complex markup is needed such as Ruby annotation (`http://www.w3.org/TR/ruby`).
RDF provides the ability to embed XML within an RDF graph instead of a string value for these cases. From a multilingual perspective, it is vital to appreciate that any language tagging must be explicitly represented within the XML or XHTML fragment as an `xml:lang` attribute value, for example, on a `<span>` or `<div>` element. Unlike with text strings within an RDF/XML document, an XML fragment ignores any `xml:lang` that lies outside it, despite the usual scoping rules for `xml:lang`. Here is a Ruby example from the OWL Test Cases:

```
<owl:Class rdf:ID="ShakespearePlay">
 <rdfs:label xml:lang="en">Shakespeare's plays</rdfs:label>
 <rdfs:label xml:lang="it">Opere di Shakespeare</rdfs:label>
 <rdfs:label rdf:parseType="Literal"
 ><span xml:lang="ja"
 >          <ruby><rbc><rb> </rb><rb> </rb></rbc>
<rtc><rt>   </rt><rt>   </rt></rtc></ruby></span></rdfs:label>
</owl:Class>
```

Notice the position of the “`ja`” language tag, differing from the position of the “`it`” tag. In particular, the following is **wrong**, and does not mark the Japanese text as Japanese:

```
<rdfs:label xml:lang="ja" rdf:parseType="Literal"
 >          <ruby><rbc><rb> </rb><rb> </rb></rbc>
<rtc><rt>   </rt><rt>   </rt></rtc></ruby></rdfs:label>
```

The `xml:lang="ja"` attribute, lies outside the literal XML content, and is ignored, due to a limitation of RDF/XML.

A further limitation of the embedded XML support is that it is primarily designed for XHTML. Certain XML applications, in particular, XSLT and XML Schema, use “Q-Names in attribute values”, which do not work with XML embedded within RDF. The namespace information, which is necessary to correctly understand such attribute values, gets lost. Hence, XSLT documents, and XML Schema documents, cannot be embedded in this way in RDF. *@@@ should this para be deleted – it does not connect with interests of multilingual computing readers.*

IRIs: Both RDF and OWL allow Internationalized Resource Identifiers to be used in place of URIs. In practice this means that almost the full range of Unicode characters can be used. This permits one string, the IRI, to be used to name the property or thing. Developers are not restricted to URIs in US ASCII only. But there is only one URI naming a property, so this does not address the multilingual need of one string for each language. Labels, as described above, are better.

Limitations with Language Tagging

RFC 3066 gives structured language tags: subtags are used for region, scripts and other things. This structure is being extended and better formalized in the current revision being worked on. Two key concepts to use with this structure are language ranges and language tag fallback. A language range is a tag with a wildcard, such as “en-*”, which covers all variations of English. Language tag fallback is a process of matching tagged resources to the linguistic abilities of an end user, by allowing the user to describe their preferences with a precise language tag, and then progressively shortening it, to provide a best match from a set of linguistic resources.

Both of these features of RFC 3066 would be useful with Semantic Web knowledge bases. Unfortunately, no support for these is routinely provided, nor envisaged in the recommendations.

In a way, the structure of language tags provide an ontology for languages on the Web, but the Web Ontology Language cannot exploit, use or interoperate with this ontology at all.

Semantic Web Query and Language

A basic function of the Semantic Web is query. The W3C is currently working on a standard query language. Right now different systems have different query languages. Mostly these query languages, like most Semantic Web software, have been developed for monolingual applications, and no support for query by language information is provided.
However, the most widely deployed query language, the RDF Data Query Language, RDQL, has recently been updated to include basic language awareness. As an example, to find the Italian label in the earlier example, you can ask the query:

```
SELECT ?resource, ?label
FROM   <http://www.w3.org/2002/03owlt/miscellaneous/consistent201>
WHERE  (?resource, rdfs:label, ?label)
AND ( ?label langeq 'it' )
```

This functionality, using the `langeq` test, is new, and will be available soon in version 2.2 of HP's Jena Semantic Web framework. Similarly functionality is expected in the query language being developed by the W3C.
In cases where the query language does not support language tagging, or when language ranges or language tag fallback are required, it is necessary to write some program code. The following snippet, shows how the API of Jena could be used to select an English label. The *matchLanguageTag* method needs to be defined, and could implement a simple case insensitive match, or a more complex wild card match for language ranges.

```
Model model = ModelFactory.createDefaultModel();
String url = "http://www.w3.org/2002/03owlt/miscellaneous/consistent201";
model.read(url);
Resource plays = model.getResource(url + "#ShakespearePlay");
StmtIterator iterator = plays.listProperties(RDFS.label);
while (iterator.hasNext()) {
   Statement triple = iterator.nextStatement();
   if ( matchLanguageTag(triple.getLanguage(),"en") )
         System.out.println(triple.getLiteral().getLexicalForm());
}
```

Code for language tag fallback would be more complex, since the best match needs to be found.

Neither of these approaches will work for the Japanese label, because the `xml:lang='ja'` is embedded within the XHTML. If necessary, the *matchLanguageTag* method could be extended to look for language information inside XML embedded within RDF.

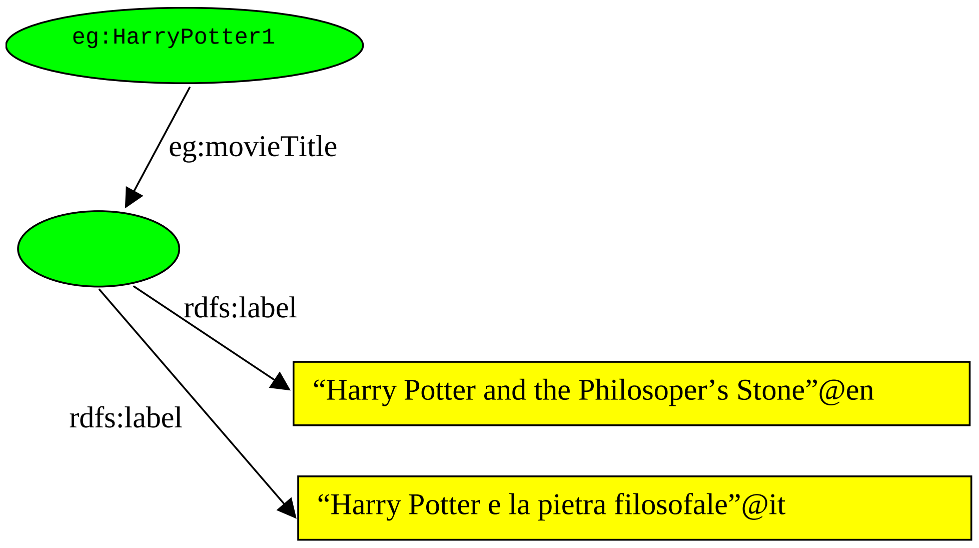
Counting and OWL

A specific problem to be faced with ontologies written in OWL concerns counting. OWL provides a variety of features that can be used to count the number of values that a property has for a particular thing. These features include `owl:FunctionalProperty`. A functional property is one which has at most one value for each distinct thing, as an example, *movie-title* is typically functional – either things like movies have one movie-title, or other things like books or Web pages, do not have a movie-title (since they are not movies).

When thinking multilingually, we see that this observation, which is true when we restrict ourselves to a monolingual view, is no longer true. A film like “Harry Potter and the Philosopher’s Stone” has many different titles, a different one for each of the languages in which the film is distributed.

This can cause specific problems when migrating an ontology developed in OWL from a monolingual application to a multilingual environment.

One approach to addressing this, is to add ‘another layer of indirection’. Specifically do not use a string as the value for a property like *movie-title*, but use an abstract resource, and give that resource many different labels, one for each language.


eg:HarryPotter1
eg:movieTitle
rdfs:label
rdfs:label
“Harry Potter and the Philosoper’s Stone”@en
“Harry Potter e la pietra filosofale”@it

The property `eg:movieTitle` is now functional, in that it has one value, but that single value can be displayed in different ways for different users.

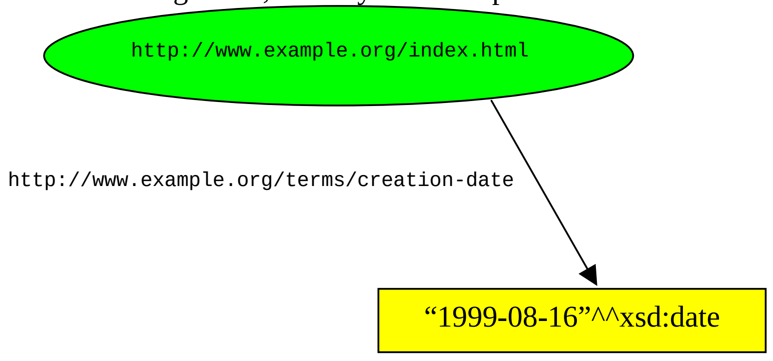
However, a refactoring of the ontology of this sort has other consequences, and you will need to work with the knowledge engineers in charge of the ontology to ensure that they are all worked through.

Locale and Semantic Web Applications

RDF and OWL provide no support for locale. The rationale is that locale is an end-user feature, and RDF and OWL are Web technologies suited for knowledge exchange across locale boundaries. Locale needs to be handled in application code.

As an example, let us consider a date. Dates can be included in RDF using the Gregorian date datatype from XML Schema, this then needs to be correctly formatted for the user locale before display.

In our knowledge base, we may have a triple like:


http://www.example.org/index.html
http://www.example.org/terms/creation-date
“1999-08-16”^^xsd:date

The value `”1999-08-16”^^xsd:date`, indicates the Gregorian date.

Java code, using Jena, that formats this date for the French locale, could include the following snippet:

```
Model model = ModelFactory.createDefaultModel();
... // read in RDF graph from Web

String url = "http://www.example.org/";
Resource index = model.getResource(url+"index.html");
Property creationDate =
            model.createProperty(url+"terms/creation-date");
// Extract date from triple store.
XSDDateTime date = (XSDDateTime)index.
                         getProperty(creationDate).
                         getLiteral().
                         getValue();
// Extract java.util.Date from XSDDateTime.
Date dateToFormat = date.asCalendar().getTime();
// Make Locale dependent formatter.
DateFormat format = DateFormat.getDateInstance(
                          DateFormat.LONG,
                          Locale.FRANCE);
System.out.println(format.format(dateToFormat));
```

Culture and Knowledge

Perhaps the hardest problem to address when internationalising a Semantic Web application is culture. The Semantic Web is used to capture knowledge, but any statement of knowledge inevitably makes cultural presuppositions, and may be inappropriate within a different culture. A simple, but pervasive, example is names. Ontologies for people developed by Americans and Europeans typically include properties such as *first-name* and *last-name*. However, different cultures name people in different ways. Representing the same knowledge as *given-name* and *family-name* is a little more robust, but still fails to be a cultural universal (Russian names for example, have a more complex structure).

Cultural issues in knowledge representation are difficult, and it is necessary to have a good knowledge of all the cultures involved to even understand what the difficulties are. No easy solution is known. Some people may believe that it is possible to represent knowledge in a culturally independent way, or depending only on features that are common to all cultures, I am deeply sceptical of such claims. Even if they are true, it remains difficult to excise the cultural dependencies from a knowledge base.

Summary: Multilingual Semantic Web Applications

These new technologies are still being tried in monolingual environments. If you are asked to help with production of a multilingual Semantic Web application you will be asking tool developers for new features, you will be pushing at the boundaries, and finding problems in the specifications - budget accordingly, try and keep multilingual features to the minimum needed (e.g. small number of languages, avoid language ranges).

Turning a monolingual system into a bilingual system should not be too difficult, by concentrating on the features highlighted above: Unicode, language tagging, labels, and embedded XML. If you hit problems, one thing you can do to help everyone is to give feedback to the W3C, so that they can continue to put *World Wide* into the Web, and the Semantic Web.

Further Reading

From the World Wide Web Consortium:

> The Recommendation concerning Resource Description Framework (Feb 2004) is split into six documents, including: *RDF Primer*, http://www.w3.org/TR/rdf-primer, edited by Frank Manola and Eric Miller; *RDF /XML Syntax*, http://www.w3.org/TR/rdf-syntax-grammar/, edited by Dave Beckett; and *RDF Concepts and Abstract Syntax*, http://www.w3.org/TR/rdf-concepts/, edited by Graham Klyne and Jeremy J. Carroll. The Recommendation for the Web Ontology Language also comes in six documents, of which the easiest is the *OWL Guide*, http://www.w3.org/TR/owl-guide/, edited by Michael K. Smith, Chris Welty, and Deborah L. McGuinness.
>
> A description of RDQL can be found in *RDQL - A Query Language for RDF* http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/ by Andy Seaborne.

Jena is available for free download from http://jena.sourceforge.net/. A high level description can be found in *Jena: Implementing the Semantic Web Recommendations* by Jeremy J. Carroll et al. available from http://www.hpl.hp.com/techreports/2003/HPL-2003-146.html

A recent draft for the next revision of RFC 3066 *Tags for Identifying Languages*, edited by A. Phillips and M. Davies, can be found at http://www.inter-locale.com/ID/draft-phillips-langtags-05.html