5 Semantic Annotation and Retrieval: Web of Hypertext – RDFa and Microformats

Ben Adida[1,4] · *Mark Birbeck*[2] · *Ivan Herman*[3]
[1]Creative Commons, San Francisco, USA
[2]WebBackplane, London, UK
[3]World Wide Web Consortium and CWI, Amsterdam, The Netherlands
[4]Harvard University, Cambridge, MA, USA

Abstract: For the Semantic Web to succeed in a major public setting, it needs to leverage the existing Web. Two major technologies have emerged over the last few years to bridge the vast, existing "clickable" Web and the new Semantic Web: Microformats and RDFa. Both allow authors to embed extra information within (X)HTML to mark up the structure, not just the visual presentation, of the information they publish. In this chapter, both approaches are explained, exploring their strengths and weaknesses, providing example applications, and touching on future considerations.

5.1   Scientific and Technical Overview

5.1.1 Introduction

Some of the introductory ideas of both the Semantic Web and RDF are presented elsewhere in this book. But a key question is how exactly might an existing Web publisher become part of the Semantic Web – in short, where does one start?

There are indeed a wide range of technologies and software packages that allow a number of sophisticated things to be done with semantic information. Ever since the start of the Semantic Web, one of the issues has been the availability of various types of data for further integration. Technically, this means making data available in RDF. One approach is to encode the RDF data in one of its serialization formats, like RDF/XML [1] or Turtle [2], but that approach requires publishers to make datasets available in a specific format whose sole purpose is the Semantic Web. Instead, interfaces to databases are being developed that can, for example, provide on-the-fly conversion of existing data into RDF, often via SPARQL [3] end points. Automatic or semiautomatic conversions exist for a number of other formats. In general it has been recognized that one should not look for one specific approach only; rather, different types of data on the Web require their own, data-specific way of expressing their content in RDF.

One of the obvious and major sources of data on the Web is, of course, hypertext, that is, HTML and XHTML. HTML is not only used by individual authors to produce Web pages directly; in fact, most of the HTML pages visible through browsers are produced by various types of back-end processes. The information itself, displayed through HTML, is often just a reflection of database content (often with significant user-interface embellishment). HTML is, therefore, a natural vehicle to incorporate the data that, ultimately, can be processed by the Semantic Web. This chapter will concentrate on how to piggyback Semantic Web data within existing hypertext, so that it can be consumed by generic Semantic Web Services.

Conveniently, the (traditional) Web community has, in parallel, come to the realization that it is beneficial to reveal more of the underlying structured information than what can simply be displayed by a browser for, essentially, human consumption. The Web 2.0 phenomenon brought a more interactive and participative Web to the fore that often relies on additional structure within the published hypertext data. Take, for example, a typical mash-up site like TripIt (http://tripit.com), which relies on the availability of data

regarding flights, weather information, or online maps. At the moment, getting to such data in an automated fashion is complicated and often involves scraping, that is, an ad hoc interpretation of various websites, or accessing site-specific APIs. Obviously, this approach does not scale to the entire Web very efficiently beyond a small handful of source websites.

This dual evolution (on the Semantic and on the more "traditional" Web) saw the emergence of a common approach (though with different techniques), namely that of (X)HTML pages including additional information that provide structural, "meta" information about what is being displayed visually. This additional information is added to the (X)HTML content as elements or attributes, providing additional information to the text. For example, if an HTML element contains the words "Ivan Herman," then an extra attribute can denote the fact that this is the name of a human being. If that extra attribute is based on some general conventions, then automated processors, reading that (X)HTML file and analyzing its structure, can automatically infer that the string is indeed a person's name and process it accordingly.

The traditional Web community has developed the Microformats approach [4] for that purpose. Microformats reuse existing HTML attributes, for example, `@class` and `@title`, and, sometimes, elements (e.g., `abbr`). In doing so, they ensure a quick and easy deployment of such annotated HTML files through existing editors and other tools. Microformat vocabularies are defined and maintained by the community; vocabularies (i.e., conventions on which elements and attributes are used and for what purpose) cover different application areas. Applications that understand a specific microformat vocabulary and know that a specific (X)HTML page abides to that vocabulary can make use of this inherent structure in the markup to get at the raw data without screen-scraping.

Although Microformats have not been developed with the Semantic Web as their target, technologies exist to convert, if necessary, XHTML files with microformat content into RDF. The transformation engine itself is, typically, XSLT [5] that converts the XHTML structure into RDF/XML (note that different XSLT scripts must be available for different Microformat vocabularies). A W3C standard, called GRDDL [6], has been defined to enable generic processors to locate the XSLT transformation for a particular Microformat, given an appropriate declaration in the XHTML content header.

Microformats provide an easy solution for simple data structures such as calendar events and contact information. However, beyond these simple cases, Microformats may become difficult to scale. The centralized Microformat definition process, which is required to ensure that one Microformat does not interfere with another (given that Microformat syntax may vary from one vocabulary to the next), significantly slows the adoption of new vocabularies, and prevents the definition of more complex vocabularies altogether. The centralized process is also required to prevent vocabulary clash: if the same HTML content contains microformat data for two different vocabularies, and both happen to use, say, the `@title` attribute, how would one decide on the intended meaning of that attribute? Finally, there is also an additional clash between the usage of the HTML attributes for Microformat purposes and their intended usage in HTML: Microformats, effectively, "hijack" certain attributes for purposes that are semantically different from their original design. As a result, Microformats may clash with existing standards or usage

patterns; for example, the `abbr` element, combined with a `@title` attribute, is often used in assistive technologies.

RDFa [7], defined by the World Wide Web Consortium (W3C), provides a more general, albeit more complex approach. Just like Microformats, RDFa relies on attributes; however, RDFa defines its own set of attributes that are used alongside existing XHTML constructs. (In contrast to some Microformat dialect, RDFa does not rely on, or introduces novel XHTML elements.) Furthermore, the values of these attributes are firmly rooted in the usage of URIs, a unique way of identifying each term; vocabulary clashes are therefore automatically avoided. This way, RDFa provides the flexibility to add essentially any information to an XHTML page, provided that a suitable vocabulary is defined that gives sense to the URIs employed by the user. All these features are, obviously, closely related to RDF and are based on the same principles. Indeed, the RDFa specification defines a precise mapping from an XHTML+RDFa file to RDF. In contrast to the Microformat+GRDDL approach, there is no need to define a separate transformation engine per vocabulary; instead, one such engine can cover all usages of XHTML+RDFa, hence the suitability of RDFa to wide-scale usage. In fact, RDFa+XHTML may be viewed as another RDF serialization format, alongside RDF/XML or Turtle, one that provides both human and machine readability. (Note, however, that RDFa does not provide syntactic shorthands for some RDF constructions like containers and collections.)

Both approaches use the structure of an (X)HTML document as a scaffold on which to attach RDF information. By using HTML in this way, RDF can be carried along wherever HTML "goes." In other words, publishing semantic information is now as easy as setting up a blog on Blogspot (http://blogspot.com), or using a content management system like Drupal. For advanced content mapped to a database, RDFa can be produced by a Rails, Struts, or ASP application on the server. In short, anyone looking to publish semantic information can use the techniques that they are already using for publishing (X)HTML.

As mentioned previously, there are already many Web applications which read data from other Web pages by examining the markup and using various heuristics to extract where the content may be. This process is often called *screen-scraping* and typically involves some stable knowledge of the target page layout, so that the proper information can be plucked from the expected position in the DOM tree. However, if the layout changes significantly, all bets are off, and the screen-scraper will need to be reconfigured. With techniques like Microformats+GRDDL or RDFa, with the RDF information embedded into the webpages, there is no need for screen-scraping, since the position and type of the information is made explicit. Hence the importance of the techniques is described in this chapter.

5.1.1.1 Applicability of Technologies

In what follows, GRDDL and RDFa will be presented only in conjunction with (X)HTML. It must be noted, however, that those technologies can also be used for other XML dialects. GRDDL has been defined in such a way that a generic processor could find a suitable XSLT transformation for any XML schema or namespaces. Similarly, the RDFa attribute set can

be added to other XML dialects; this has already been done for SVG Tiny 1.2 [9] or Yahoo!'s MediaRSS [10].

5.1.2 Microformats and the Semantic Web

(See also ❯ *Table 5.1* for an overview on Microformats and GRDDL.)

5.1.2.1 Microformats

Microformats reuse existing attributes in the HTML specification to add semantic information to the content. The term "Microformats" does not refer to one specific vocabulary, but rather to a general approach of using these attributes. Each application is assigned its own vocabulary and associated syntax. For example, hCard is used to describe people and organizations, hCalendar to publish calendar data, and hAtom for news feeds. The community around the http://www.microformats.org site is set up to shepherd the process of developing and standardizing new Microformats. These Microformat vocabularies are rarely defined from scratch; instead, they are a translation of existing vocabularies to the Microformats approach (e.g., hCalendar is a translation of vCalendar, hAtom of ATOM, hCard for vCard, etc.).

Technically, the semantic information itself is encoded in attribute values using simple strings. The definition of a new Microformat consists of defining these strings and giving a textual description as to their intended meaning. For example, the hCalendar specification [10] specifies the meaning of "dtstart" (the starting date of a calendar event), "location" (the location of a calendar event), or "geo" (the latitude and longitude values of the location). The attribute of choice to encode these values is the `@class` attribute.

As a very simple example, the following HTML:

```
<div class="vevent">
   <a href="http://2008.sxsw.com/interactive/" class="url">
    <span class="summary">SXSW Interactive</span>
   </a>
</div>
```

◘ **Table 5.1**

Common problems and their solutions using Microformats + GRDDL

Problem	Solution in Microformat + GRDDL
Define terms in Microformats	Define attribute values, usually for `@class`, (sometimes for `@title` or the `@abbr` element)
Convert terms to RDF	Apply a GRDDL transformation (usually XSLT)
Find GRDDL transform (general)	Use `@profile` with generic value and a `<link>` element with `@rel="transformation"`
Find GRDDL transform (for specific, frequent cases)	Use `@profile` with a specific value (leading to another document to be GRDDL-d)

provides the information that (1) this is an event, (2) it can be summarized as "SXSW Interactive," and (3) it has a homepage (the usual approach is to use the `@href` attribute value as the URI corresponding to the "url" tag).

It is, in some cases, not possible to rely exclusively on `@class` and the existing HTML content structure; extra elements and possibly other attributes should also be used. A typical example, in hCalendar, is the setting of dates, which usually relies on the `abbr` element and uses the `@title` attribute for the precise (i.e., ISO formatted) date value instead of the human-readable form. So, for example, the code above could be extended by adding the start and end dates of the event as follows:

```
<div class="vevent">
  <abbr class="dtstart" title="2008-03-07">Mar 7</abbr>-
  <abbr class="dtend" title="2008-03-12">11, 2008</abbr>:
  <a href="http://2008.sxsw.com/interactive/" class="url">
    <span class="summary">SXSW Interactive</span>
  </a>
</div>
```

❯ *Figure 5.1* is a typical example for using Microformats. The page uses two different Microformats (hCard and hCalendar) in an otherwise typical Web page, denoting the
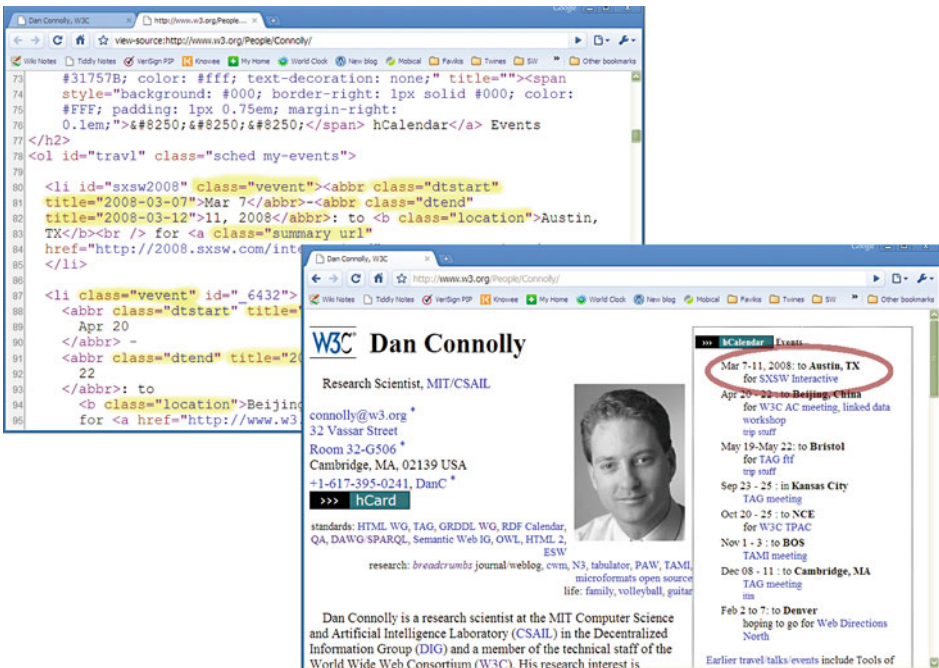
▣ Fig. 5.1
Typical example of hCalendar usage

address data and conference events of the person. The extra information is embedded in the XHTML page but those extra attributes are not visible on the browser screen. The page reads just like any other page. The figure also shows what is behind the screen, with some of the hCalendar tags highlighted. Tools exist to extract, for example, iCalendar data from a hCalendar Microformat markup, yielding (for this example):

```
BEGIN:VCALENDAR
...
BEGIN:VEVENT
URL:http://2008.sxsw.com/interactive/
LOCATION;CHARSET=UTF-8:Austin\, TX
SUMMARY;CHARSET=UTF-8:SXSW Interactive
DTSTART:20080307
DTEND:20080312
END:VEVENT
```

Microformats are usually registered on the community site [4], after discussions on dedicated mailing lists, blogs, or IRC channels. In this sense, it is a very typical case of a "grassroots" effort. The site repertories Microformat specifications, both in final and draft formats. These include the hCalendar and hCard specifications used in the examples, but also Microformats for audio content, simple voting procedures, or for publishing resumes and CVs.

Microformats provide a quick and effective way to add a little bit of semantics to HTML pages. The fact that they reuse existing HTML attributes and elements also means that it is fairly easy to edit HTML pages with Microformats (using various HTML editors) and adding the microformat attribute values does not endanger the validity of the page with the usual HTML DTDs and validators. The relative simplicity of the terms makes it also easy to deploy Microformats.

Microformats have some drawbacks, however. The main issue is that the semantic meaning described in the microformat tags are all defined in isolation, independently of one another. Even if the same HTML file contains different Microformats (as in the example on ❯ *Fig. 5.1*), developing an application that integrates the data from those two tag-sets means a case-by-case development in understanding what all the different tags stand for, how they are encoded, what parsing procedures should be followed, etc. This integration task may be made more complicated by a possible clash in the microformat specifications; what happens if two different Microformat dialects use the same attribute value or the same HTML element for different purposes? Typically, the Microformats community ensures that no conflicts between different Microformats arise, resulting in sometimes confusing semantic choices for simple terms, for example, "title" means "Job Title" because hResume was the first Microformat to lay claim to it. However, the centralized control to prevent conflicts means that either no decentralized innovation will occur, or name collisions will.

5.1.2.2 **Transforming Microformats to RDF**

Further complications arise with the usage of Microformats if the encoded semantic meaning is to be integrated with different sources of data that are not necessarily available in an HTML file. This integration aspect is the central paradigm of the Semantic Web, with RDF [11] playing the role of a glue among different data schemas. As a consequence, a natural way of integrating Microformats into the Semantic Web – while still keeping their benefits – is to extract the semantic content and convert it into RDF.

If the host language is XHTML, that is, a dialect of XML, then the straightforward way of doing that is to use a suitable XSLT [5] transformation. XSLT is a functional style programming language, itself encoded in XML, which describes the transformation of an XML tree into either another XML tree or simply into plain text. Since its inception, XLST has become an integral part of the XML infrastructure. Implementations of XSLT processors are widely available as stand-alone programs and as part of XML editors or Web browsers. Due to the simplicity of the usual Microformat dialects, the definition of an XSLT transformation to extract the semantic data and generate an RDF equivalent in RDF/XML (or Turtle) is usually a relatively straightforward task. Here is an example of the RDF output an XSLT processor may produce for the example above:

```
<http://www.w3.org/People/Connolly/#sxsw2008>
   a <http://www.w3.org/2002/12/cal/icaltzd#Vevent>;
   <http://www.w3.org/2002/12/cal/icaltzd#summary>
      "SXSW Interactive";
   <http://www.w3.org/2002/12/cal/icaltzd#dtstart>
      "2008-03-07"^^xsd:date;
   <http://www.w3.org/2002/12/cal/icaltzd#dtend>
      "2008-03-12"^^xsd:date;
   <http://www.w3.org/2002/12/cal/icaltzd#url>
      <http://2008.sxsw.com/interactive/>;
   <http://www.w3.org/2002/12/cal/icaltzd#location>
      "Austin, TX".
```

The transformation depends on an RDF vocabulary that is equivalent to the Microformat dialect (the `.../icaltzd` namespace in this example). Thus, to transform the Microformat data into RDF, thereby integrating with other types of data on the Web, two steps have to be followed:

1. Define an RDF vocabulary corresponding to the Microformat dialect.
2. Create an XSLT transformation that extracts the data from the XHTML source and produces RDF using that vocabulary.

A number of XSLT transformations have already been defined and published for the more widely used Microformats. A wiki site has been set up at the W3C [12] listing some of these, including references to the RDF vocabularies as well as to the XSLT transformations

themselves. The site is maintained by the community at large, and is not under the official control of any W3C or other groups.

5.1.2.3 GRDDL

The XSLT transformation approach described in the previous section does not cover all what is needed to smoothly integrate Microformats with the Semantic Web. Indeed, even though transformations may be defined, the issue of finding the appropriate transformation remains. When a processor finds an XHTML page containing a Microformat, how does that processor know which transformation to apply? By default, the Microformat dialect itself does not provide an answer.

This is where an additional W3C technology, called GRDDL [6], comes in. GRDDL uses existing (X)HTML constructs to locate the necessary transformation(s) for the HTML markup. A generic GRDDL processor would then find and download the transformation (at least conceptually, because an optimized GRDDL processor would probably cache the transformation for all major Microformats), execute the transformation, and produce the RDF output. This means that fully generic and automatic GRDDL processors can be created and added to general RDF application environments. Put another way, a user can point at a URI of a GRDDL-d page annotated with Microformats just as if it was a URI referring to an RDF/XML file.

What are the additional attributes defined by GRDDL? The simplest way of defining the GRDDL transformation is to modify the header of an XHTML file as follows:

– Add a @profile attribute to the XHTML head denoting the fact that the XHTML file is to be processed by a GRDDL processor; and use a "link" element in the head to indicate the URI of the transformation.

For example, the previous example involving the hCalendar Microformat could have the following header:

```
<html>
  <head profile="http://www.w3.org/2003/g/data-view">
    <link rel="transformation"
      href="http://www.w3.org/2002/12/cal/glean-hcal.xsl"/>
    ...
  </head>
...
</html>
```

If the page uses several Microformats, separate link elements should be added for each of those. The GRDDL processor would then execute each transformation separately and perform an RDF merge on the generated graphs (as described in the RDF Semantics document [11]), yielding the final result.

GRDDL also defines a way to add the necessary attributes to a dialect of XHTML files, thereby simplifying the task of authors. Whereas the profile value http://www.w3.org/ 2003/g/data-view simply means "process this file through a GRDDL processor," publishers can also use dialect-specific @profile values. The goal of these specific values is to "merge" the two items of information above, that is, making the separate link element unnecessary.

Technically, what happens is as follows:

1. The GRDDL processor follows this special profile URI, retrieves the target (i.e., profile) document.
2. Performs a GRDDL transform on the profile document, yielding a (small) RDF graph.
3. This RDF graph contains a reference to another transformation.
4. This transformation is applied on the original source.

One could say that the usage of the dialect-specific profile leads to some sort of an indirect application of GRDDL.

For example, the file at the URI http://www.w3.org/2002/12/cal/hcal contains the necessary GRDDL information to yield the RDF triple (the predicate URI is defined by the GRDDL standard):

```
<> <http://www.w3.org/2003/g/data-view#profileTransformation>
      <http://www.w3.org/2002/12/cal/glean-hcal.xsl>.
```

The object of this triple is identical to the transformation provided by the simple GRDDL markup for hCalendar. The XHTML source of the example can therefore be modified to (note the removal of the link element):

```
<html>
   <head profile="http://www.w3.org/2002/12/cal/hcal">
   </head>
   ...
</html>
```

Through the indirect step, the GRDDL processor will produce the same RDF graph as before.

As mentioned above, many RDF environments, for example, Redland (http://librdf. org) or Jena (http://jena.sourceforge.net/), or RDF browsers like the Tabulator (http:// www.w3.org/2005/ajar/tab) already include automatic GRDDL processors in their newer releases. What this means in practice is that XHTML files with Microformats and with the necessary GRDDL attribute values will be processed, and the resulting RDF graph(s) added to, for example, a SPARQL query dataset automatically.

As already referred to in the introduction, the attributes defined by the GRDDL recommendation have their purely XML variants that do not depend on the XHTML structure (i.e., the html and link elements). The indirect approach is then achieved through the namespace document instead of the profile document. This turns GRDDL into an extremely powerful technology to build a bridge between the XML world and the

RDF world in general. The interested reader should refer to the GRDDL Recommendation [6] or the GRDDL Primer [13] for further details.

5.1.3 RDFa

RDFa shares a number of high-level design principles with Microformats. Both use (X)HTML attributes to express the additional data that can be retrieved from the (X)HMTL content and converted into RDF, and both rely on external processing to retrieve the necessary information by interpreting the structure of the document. RDFa is, however, more flexible: if an RDF vocabulary has been defined by a community for some purpose, that vocabulary can be directly reused by RDFa publishers, whereas Microformats must redefine that vocabulary, through some additional community agreement, into a microformat dialect.

As an example (used later in this section), a widely used vocabulary in RDF applications is based on FOAF [14] (Friend-of-a-Friend). FOAF is used to describe personal data (names, phone numbers, etc.) as well as the relationships between people (e.g., that one person knows another). By publishing FOAF files about themselves, people can export, for example, their social relationships independently of a particular social network application. In practice, personal profile files use a number of different vocabularies beyond FOAF; indeed, the FOAF vocabulary itself tries to be minimalistic and relies on the usage of other terminologies to express, for example, geographical coordinates or contact addresses. Mixing various terminologies in the FOAF context is no problem in RDF – after all, integrating various types of vocabularies is one of the main advantages of using it.

FOAF files can be created by various tools or indeed created directly in, say, a text editor using an RDF serialization like Turtle. However, one has to realize that the data put into a FOAF file are almost identical to what one would put into one's CV: name, surname, schools, personal interests, e-mail addresses, etc. It is therefore a fairly obvious wish to merge these two; one should write a single XHTML file to publish a CV, and RDF applications should be able to extract the FOAF information automatically for further integration with other types of data.

Using Microformats to achieve this does not really work. First of all, the FOAF vocabulary is relatively large, which means that a hypothetical Microformat version, as well as a necessary GRDDL/XSLT transform, would become fairly complex. But, and perhaps more importantly, the fact that FOAF files typically mix many different vocabularies would make it unmanageable, if not outright impossible, for Microformats to produce CVs in XHTML that could also be used to automatically generate FOAF data (❯ *Fig. 5.2*).

All this is not a problem for RDFa. Indeed, RDFa is perfectly well prepared to handle several vocabularies within one file. This is because RDFa is based on the usage of URIs, meaning that usage of various terms can be interleaved without any particular problems. Because vocabularies defined on the Semantic Web are also based on URIs, this means that RDFa can readily incorporate and benefit of the existence of many vocabularies and ontologies that the Semantic Web community develops, regardless of the complexity of these.

To make the usage of URIs easier, RDFa employs a namespace-like mechanism to abbreviate URIs via strings like dc:title or foaf:Person to replace full URIs (those
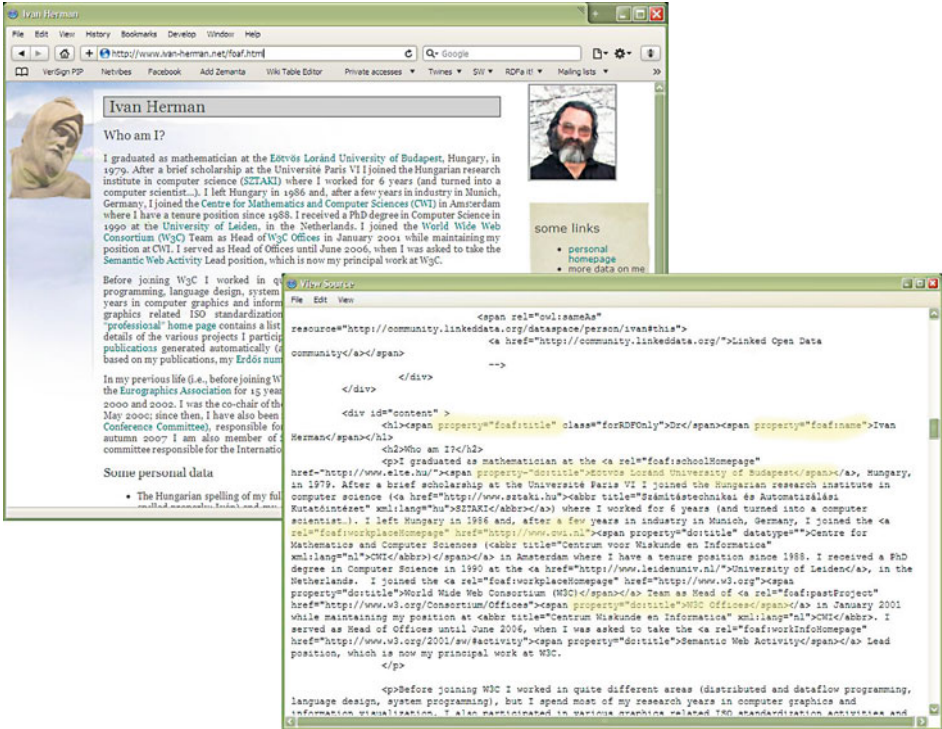
◼ Fig. 5.2
CV file in XHTML with RDFa markup

URIs can be confusingly long). Users can declare these prefixes using the `xmlns:` attributes on any element in the XHTML hierarchy. These "compact URIs," or CURIEs, are used in specific attributes, such as `@rel` and `@property`. When an attribute needs to use either a URI or a CURIE, for example in `@about`, CURIEs are differentiated using square brackets, for example, `[prefix:suffix]`. Those cases are, however, rare.

5.1.3.1 Adding Triples with Literals

So how does RDFa work? This section will use the example of FOAF files to introduce the various RDFa constructs. The specification of RDFa is defined in terms of RDF; essentially, what it does is to define what kind of RDF triples a specific combination of XHTML elements and attributes yield (noting that the attributes may be RDFa specific or, in some cases, native XHTML ones). This section will follow the same approach. (See also for an overview of the RDFa constructs presented here.)

One obvious entry in a CV file is one's name. The XHTML file may look something like:

```
<h1>Ivan Herman</h1>
```

An RDFa attribute can be added to this statement:

◘ **Table 5.2**

Common problems and their solutions using RDFa

Problem	Solution in RDFa
Abbreviate long URIs	`@xmlns` and CURIEs
Define a predicate with a literal object	`@profile` and the text content of the element
Define a predicate with a literal object (alternative)	`@profile` and `@content`
Set language tag of a literal	`Qxml:lang`
Set datatype of a literal	`@datatype`
Define a predicate with a URI resource object	`@rel`
Define a URI resource object	`@href` or `@resource`
Define the subject	`@about` (or inherit the value of `@resource` from the hierarchy)
Set type of a resource (`rdf:type`)	`typeof`
Define a blank node	`_:xx` style CURIE, or `@typeof` without a `@resource` or `@href` on the element

```
<h1 property="foaf:name">Ivan Herman</h1>
```

The extra `@property` instructs an RDFa-aware tool that whatever is enclosed in an element should be taken verbatim as literal, that is, to generate the following triple:

```
<> foaf:name "Ivan Herman".
```

Both in the XHTML/RDFa encoding and the final triple the term `foaf:name` stands for the abbreviation of a URI of the form:

```
http://xmlns.com/foaf/0.1/name
```

The pairing of the `foaf` prefix and its corresponding URI is done, in the case of RDFa, using an attribute declaration of the form:

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

somewhere in the XHTML hierarchy. (Sometimes, for convenience in templating systems, it will be placed at the top, that is, on the `html` element, but it can also be elsewhere.)

The example above defines a triple with a literal object, where the literal is automatically drawn from the XHTML text. Such a literal, in some cases, might be in another language than English. For example, the same file can also include the following:

```
The Hungarian spelling of my full name is
<span property="foaf:name" xml:lang="hu">Herman Iván</span>
```

that yields the following triple:

```
<> foaf:name "Herman Iván"@hu.
```

which is the Turtle notation to denote an RDF literal with a language tag. Note that RDFa does not introduce a separate attribute to denote a language; instead, the standard XHTML attribute `xml:lang` is reused. (The current version of RDFa does not use the `@lang` attribute, because that was not used by XHTML1.1 at the time of the specification. That has recently changed, so future versions of RDFa will also allow the usage `@lang`.)

Literals can also have datatypes, like floats, dates, integers, etc. For example, one could write:

```
<span property="foaf:birthday"
   datatype="xsd:date">1955-02-24</span>
```

leading to:

```
<> foaf:birthday "1955-02-24"^^xsd:date.
```

using, again, the Turtle notation to denote a triple with an RDF Literal object with a datatype. RDFa is silent on what kind of datatypes can be used, the value of `@datatype` is simply a (compact) URI. In practice, in accordance with the vast majority of RDF applications, the XSD datatypes [15] are used.

The last example also shows that, in some cases, the approach of relying on the XHTML text for literals is not optimal. Indeed, though the term `1955-02-24` is the standard way of denoting a date, it does look slightly unnatural in an English text. RDFa, therefore, introduces the `@content` attribute that can be used to denote a literal that is not part of the XHTML text. The example above could have been written as:

```
<span property="foaf:birthday" content="1955-02-24"
   datatype="xsd:date">24th February, 1955</span>
```

yielding exactly the same RDF triple but making the XHTML text more readable.

Speaking of dates, it is usually a good practice to add a date to the page, signaling the last update of the CV (as well as the generated FOAF data). This can easily be added to the page, using:

```
<span property="dc:date" datatype="xsd:date">2009-03-30</span>
```

The only difference is the usage of a different vocabulary, in this case a so-called Dublin Core [16] term. Adding this to the RDFa file is easy; the only requirement is to add another `xmlns:` attribute somewhere defining the URI for the `dc:` prefix; from that point on the FOAF and Dublin Core properties can be intermixed within the same RDFa file, and the RDFa processor would automatically generate the right triples. The same mechanism can be used with any number of vocabularies making it possible to mix several vocabularies within one XHTML/RDFa file easily.

`@property` attributes can list several values, in case triples with identical subjects and objects but with different predicates are to be generated. For example, it is possible to say:

```
<span property="dc:date dc:created"
   datatype="xsd:date">2009-03-30</span>
```

yielding two triples instead of one:

```
<> dc:date "2009-03-30"^^xsd:date.
<> dc:created "2009-03-30"^^xsd:date.
```

5.1.3.2 Nonliteral Objects

Of course, not all objects in triples are literals. For example, the FOAF vocabulary includes a term called `foaf:workplaceHomepage`. The object of a triple using this predicate should be a URI resource rather than a Literal. To differentiate between the two types of possible objects, RDFa uses the existing `@rel` and `@href` attributes for nonliteral objects. Using these the code

```
I joined the <a rel="foaf:workplaceHomepage"
    href=http://www.w3.org> Wide Web Consortium (W3C)</a>
```

yields

```
<> foaf:workplaceHomepage <http://www.w3.org>.
```

The `@rel` attribute is not specific to RDFa; it is a valid XHTML attribute although, in traditional XHTML, its usage is restricted to the `link` and `a` elements. RDFa extends its usage to any element. (Conceptually, the RDFa usage of `@rel` is compatible with the original interpretation in XHTML, although expressed in RDF terms.)

In some cases, especially when a clickable anchor element is used, the intended RDF object is not exactly the same as the clickable link. This is similar to the case where `@content` is used to "override" the human-readable data for machine-readability purposes. When it comes to URI objects, the method for such an override is to use `@resource` instead of `@href`. When both are used, only `@resource` is considered for RDFa purposes. Thus, it becomes possible, when a resource has two different URIs for machine and human readability, to link to both, one for each audience.

It is also possible to use both the `@rel` and the `@property` attributes on the same element; since their interpretation does not collide, it may be a succinct way of expressing several triples, one that considers the link target as a URI object, and another that considers the anchor text as a literal object. For example, something like

```
<span property="dc:title" rel="foaf:workplaceHomepage"
    resource="http://www.w3.org">
        World Wide Web Consortium (W3C)</span>
```

yields

```
<> foaf:workplaceHomepage <http://www.w3.org>.
<> dc:title "World Wide Web Consortium (W3C)".
```

Note that this version of the code uses the `span` element and the `@resource` attribute, meaning that the element is not clickable on the browser screen. The same triple is generated nevertheless.

5.1.3.3   Specifying Subjects

Up until this point all generated triples used `<>` as subjects, denoting the resource with the base URI. While this is fine for simple cases, it may not work for more complex examples. To address this issue, RDFa introduces the `@about` attribute. This attribute (reminiscent of `@rdf:about` in RDF/XML) makes a subject explicit. Specifically, the very first example would be, more realistically:

```
<h1 about="http://www.ivan-herman.net/foaf#me"
   property="foaf:name">Ivan Herman</h1>
```

generating:

```
<http://www.ivan-herman.net/foaf#me> foaf:name "Ivan Herman".
```

An important aspect of `@about` is that it does not only specify the subject for the element where it appears, but for all descendants, too (unless it is overwritten by another `@about`). In other words, the following, slightly more complex RDFa code:

```
<div about="http://www.ivan-herman.net/foaf#me">
   <h1 property="foaf:name">Ivan Herman</h1>
   <ul>
     <li><a rel="foaf:homepage"
         href="http://www.ivan-herman.net">personal homage</a></li>
     <li><a rel="foaf:weblog"
         href="http://www.ivan-herman.name" personal blog</a></li>
   ...
```

generates the following triples:

```
<http://www.ivan-herman.net/foaf#me> foaf:name
   "Ivan Herman" .
<http://www.ivan-herman.net/foaf#me> foaf:homepage
   <http://www.ivan-herman.net> .
<http://www.ivan-herman.net/foaf#me> foaf:weblog
   <http://www.ivan-herman.name>.
```

In other words, using `@about` makes it possible to group a number of triples sharing the same subject. Note that this is not unlike the Turtle idiom of regrouping

triples using the ';' separator; indeed, the previous Turtle triples could have been written as:

```
<http://www.ivan-herman.net/foaf#me>
   foaf:name "Ivan Herman" ;
   foaf:homepage <http://www.ivan-herman.net> ;
   foaf:weblog <http://www.ivan-herman.name>.
```

which does show similarities to the structure of the XHTML code. (RDF/XML has similar simplification possibilities.)

Using @about is not the only way to set the subject. Another approach is to make use of nesting in RDFa. Nesting means that a @href (or @resource) attribute not only sets the object of a triple but also acts as an implicit @about for all descendant nodes. The easiest way to understand that is again in an example. Consider the following:

```
<div about="http://www.ivan-herman.net/foaf#me">
...
   I joined the <a rel="foaf:workplaceHomepage"
     href="http://www.w3.org"> Wide Web Consortium (W3C)</a>
...
```

which yields, as before:

```
<http://www.ivan-herman.net/foaf#me>
   foaf:workplaceHomage <http://www.w3.org>.
```

However, one may want to add some additional triples with http://www.w3.org as subjects, for example, using the (Dublin Core) dc:title predicate to denote the title of a resource. The nesting feature of RDFa makes it possible as follows:

```
<div about="http://www.ivan-herman.net/foaf#me">
...
   I joined the <a rel="foaf:workplaceHomepage"
     href="http://www.w3.org">
       <span property="dc:title">World Wide Web Consortium (W3C)
       </span> </a>
...
```

The span element is nested as a child element of a; using the nesting rule of RDFa means that the following triples will be generated:

```
<http://www.ivan-herman.net/foaf#me>
   foaf:workplaceHomage <http://www.w3.org> .
<http://www.w3.org>
   dc:title "World Wide Web Consortium (W3C)" .
```

5.1.3.4 Grouping Objects

As already noted, `@about` can be used to group triples that share the same subject. Another frequent situation in practice is when a number of triples share not only the subject but the subject and the predicate. Consider the following situation in RDFa:

```
<div about="http://www.ivan-herman.net/foaf#me">
...
   <ul>
     <li>
       <a rel="foaf:holdsAccount"
          href="http://www.dopplr.com/traveller/IvanHerman">
          dopplr
       </a>
     </li>
     <li>
       <a rel="foaf:holdsAccount"
          href="http://www.facebook.com/ivan.herman">
          Facebook
       </a>
     </li>
...
```

using the definition of `@about` this code yields:

```
<http://www.ivan-herman.net/foaf#me>
   foaf:holdsAccount
     <http://www.dopplr.com/traveller/IvanHerman> .
   <http://www.ivan-herman.net/foaf#me>
     foaf:holdsAccount
      <http://www.facebook.com/ivan.herman> .
```

However, RDFa provides the ability to take the common predicate (the `@rel` value) out into an enclosing element to avoid repeating it. The RDFa code could have been written as:

```
<div about="http://www.ivan-herman.net/foaf#me">
...
   <ul rel="foaf:holdsAccount" >
     <li>
       <a href="http://www.dopplr.com/traveller/IvanHerman">
          dopplr
       </a>
     </li>
```

```
<li>
  <a href="http://www.facebook.com/ivan.herman">
     Facebook
  </a>
</li>
...
```

generating the very same set of triples, but in a more economical (and therefore less error-prone) way. This is, again, not unlike the simplification possibilities offered by Turtle; indeed, the same triples could have been written as:

```
<http://www.ivan-herman.net/foaf#me>
   foaf:holdsAccount
     <http://www.dopplr.com/traveller/IvanHerman>,
     <http://www.facebook.com/ivan.herman> .
```

5.1.3.5 Typing

One of the important patterns in RDF is the usage of RDFS or OWL classes with typing. For example, the FOAF vocabulary includes the notion of a `Person` and, usually, the FOAF data themselves contain a triple of the form:

```
<http://www.ivan-herman.net/foaf#me> rdf:type foaf:Person.
```

This can be expressed in RDFa using the `rdf:type` predicate explicitly, for example,

```
<h1 about="http://www.ivan-herman.net/foaf#me"
   rel="rdf:type" resource="[foaf:Person]">Ivan Herman</h1>
```

would generate the triple above. (Remember that the [and] notation is used to denote a CURIE when the attribute value must otherwise be a URI. As `@resource` is close to `@href` and the value of `@href` is defined as URI in XHTML, `@resource` inherits this restriction.)

Because this typing triple is a recurring pattern in RDF and, therefore, in RDFa, a special attribute called `@typeof` is introduced to make this easier. Using this attribute, the XHTML code above could be rewritten as:

```
<h1 about="http://www.ivan-herman.net/foaf#me"
    typeof="foaf:Person">Ivan Herman</h1>
```

One of the advantages of using `@typeof` is that it becomes easier to combine typing with other RDFa attributes. For example:

```
<h1 about="http://www.ivan-herman.net/foaf#me"
    typeof="foaf:Person" property="foaf:name">Ivan Herman</h1>
```

succinctly generates two triples, namely:

```
<http://www.ivan-herman.net/foaf#me>
   rdf:type foaf:Person ;
   foaf:name "Ivan Herman" .
```

5.1.3.6   Local (Blank) Nodes

Yet another important concept of RDF is blank nodes. It is not necessary to go into the precise mathematical specification of blanks nodes here; suffice it to say that blank nodes represent local nodes in an RDF graph that are not merged with any other nodes in another graph even if their names happen to coincide. (RDF environments are supposed to perform an internal renaming of the nodes in such cases.)

RDFa allows the usage of the _:x symbols (where x can be any string) to denote a blank node wherever a compact URI (CURIE) is used. This is, again, similar to the Turtle serialization that uses a similar syntax to denote blank nodes. For example, the following RDFa code:

```
<ul about="http://www.ivan-herman.net/foaf#me" rel="foaf:knows">
   <li resource="[_:a]" typeof="foaf:Person">
      <span property="foaf:name">Ben Adida</span>
   </li>
```

(Note again the [and] that surround the CURIE in @resource to make it distinct from a "real" URI.) The generated triples, in Turtle, may be an almost verbatim translation:

```
<http://www.ivan-herman.net/foaf#me> foaf:knows _:a .
_:a rdf:type foaf:Person;
    foaf:name "Ben Adida" .
```

The RDFa code can include several blank node names, something like:

```
<ul about="http://www.ivan-herman.net/foaf#me" rel="foaf:knows" >
   <li resource="[_:a]" typeof="foaf:Person">
     <span property="foaf:name">Ben Adida</span>
   </li>
   <li resource="[_:b]" typeof="foaf:Person">
     <span property="foaf:name">Mark Birbeck</span>
   </li>
```

yielding two different blank nodes. This type of pattern is fairly frequent, so RDFa includes yet another extra rule regarding the @typeof attribute that makes the generation of such graphs easier. Although the exact rules are a little bit complicated, the essence is that by simply removing @resource (which is used to generate an internal identifier anyway) one could write:

```
<ul about="http://www.ivan-herman.net/foaf#me" rel="foaf:knows" >
   <li typeof="foaf:Person">
      <span property="foaf:name">Ben Adida</span>
   </li>
   <li typeof="foaf:Person">
      <span property="foaf:name">Mark Birbeck</span>
   </li>
```

This encodes exactly the same RDF data: the `@typeof` attribute without the `@resource` (or `@href`) will automatically generate its own, local, blank node. Note, again, the similarity to a possible, more compact Turtle encoding of these triples:

```
<http://www.ivan-herman.net/foaf#me> foaf:knows
    [a foaf:Person; foaf:name "Ben Adida"],
    [a foaf:Person; foaf:name "Mark Birbeck"].
```

Although the full RDFa specification includes some more details not covered here (on the more complex behavior of `@typeof`, `@datatype`, generation of so-called XML literals, etc.), this chapter should provide a good enough overview of what can be achieved with RDFa in combination with XHTML. For further details, the interested reader should refer to the RDFa Recommendation [7] or the RDFa Primer [17].

5.1.4 GRDDL and RDFa

This chapter presented GRDDL and RDFa as if they were completely independent technologies. However, this is not exactly the case.

It has already been emphasized that GRDDL is not "bound" to Microformats (nor to XHTML, in fact): although Microformats provide us with probably the most important use case for GRDDL, the specification itself is more general.

It provides a standard way to locate a suitable transformation that could be used to transform the underlying XML (i.e., including XHTML) data into RDF. The specification is silent on what that script does exactly, provided it produces proper RDF. This means that it is perfectly possible to use GRDDL to extract triples from an RDFa+XHTML file: all that is needed is to have a suitable transformation available and to add the necessary markup to the RDFa+XHTML file to use this transformation via GRDDL.

Such XSLT transformation does indeed exist; users wishing to use it may do so by adding the necessary GRDDL markup to their RDFa+XHTML file (see [18] for details). Actually, even this may not be necessary: indeed, the namespace document for XHTML (http://www.w3.org/1999/xhtml/) is already prepared for indirect GRDDL processing (and the XHTML namespace is usually present on the `html` element of XHTML files). Truth must be said, however, that at the time of finalizing this manuscript, not many GRDDL implementations implement the indirect approach. That is, using the simple GRDDL markup, as described in [18], might be safer for the time being.

5.2 **Example Applications**

Microformats have achieved notable use in a number of "Web 2.0" applications, including supporters such as Microsoft and Apple. RDFa, though more complex, has also achieved use in a number of particularly exciting scenarios. Oftentimes, applications choose to support both Microformats and RDFa when the vocabularies are simple enough to allow it. This chapter gives some examples.

5.2.1 **Connections Between the Human and Machine-Readable Web**

As already referred to in the introduction, one of the main issues for the Semantic Web (or a "Web of Data" in this context) is the availability of data. Beyond the more traditional sources of information (on-the-fly access to databases, XML files, etc.) one of the most important sources is the vibrant world of Web pages, with the dynamic nature provided by such typical Web 2.0 tools as blogs, microblogs, social sites, etc. However, bridging the gap between the "traditional" and the Semantic Web has always been a major source of controversy and difficulties. Indeed, it is difficult, if not impossible, to expect website authors, bloggers, contributors to various Web 2.0 sites, or even automated tools to produce, alongside the (X)HTML pages, separate metadata in a different format like RDF/XML or Turtle. The barriers are not only technical, but also "social": many website designers, content providers, etc., do not have clear notions of such concepts as metadata, vocabularies, etc. This in spite of the fact that new applications (such as the ones described in more details later in this section) provide very tangible values based on the usage of such additional data.

Both Microformats and RDFa have a major role to play in solving this problem. They offer technologies whereby the same source of information can be used both by the human and the machine; it offers new perspectives to technologies that can generate such additional information either automatically or with a very little help from the users. More importantly, they offer a very easy transition path for existing tools to provide richer content; there is no need for the generation of separate files on a website, separate entries in the database of a CMS system, etc.

A good example of this evolution is a CMS system like Drupal. This open-source tool has helped many organizations to build their online presence and publish content on the Web, such as the White House or the World Bank. With an estimated half million websites running on it, Drupal is ranked among the first three CMS platforms in use today.

Drupal (http://www.drupal.org) has recently incorporated Semantic Web structures in general, and RDFa in particular, into its core engine. Drupal 7's internal data structures are mapped to RDF by default and its theme layer – responsible for rendering the HTML elements – injects RDFa markup in the Web page. Fields exposed by this RDFa are title, date of publication, author information, main content, as well as the possible comments

on the page [19]. This is done automatically for all users and pages without any prior knowledge of RDFa (just as the average user does not know HTML markup). Other CMS services (like Liferay, http://www.liferay.com/) are also looking at Semantic Web technologies and, in particular, at RDF to add additional meta information to their Web pages, and it can be expected that some sort of microformat or RDFa will be used by those, too.

Another automatic source of adding microformat and/or RDFa information directly into the pages are plug-ins or additional tools that can be used in conjunction with, say, blogging environments. For example, a plug-in exists for the popular WordPress platform (http://wordpress.org/extend/plugins/wp-rdfa/) to automatically add FOAF Dublin Core tags to blog entries using RDFa. The services provided by tools like Zemanta (http://www.zemanta.com), Open Calais' Marmoset (http://www.opencalais.com/documentation/calais-marmoset), or Nstein (http://www.nstein.com) go one step further: these tools analyze the textual content of, for example, a blog entry and propose common tags (based, e.g., on DBpedia terms) by adding microformat or RDFa to the content. These tools can be used either via dedicated APIs or, for example, through plug-ins to blogging environments like WordPress, Movable Type, or even CMS systems like Drupal. (It is interesting to note that there is even a recent initiative to agree on a common tagging format using RDFa, see http://commontag.org/Home.)

Tools like Open Calais or Zemanta rely on fairly complex natural language processing techniques. The appearance of Microformats and of RDFa has provided a long-awaited mechanism, whereby the results of such technologies can be finally exploited on the Semantic Web. To put it succinctly, Microformats and RDFa contribute in making the gap between the human and the machine-readable Web eventually disappear.

5.2.2 Search

The Semantic Web has long proposed the idea that, with explicit semantics, search engines may become smarter. With Microformats and RDFa, this evolution is beginning.

5.2.2.1   Yahoo! SearchMonkey and BOSS

In the summer of 2008 Yahoo! released SearchMonkey (http://developer.yahoo.com/searchmonkey/) [20, 21], an API that enables developers to customize the presentation of Yahoo!'s search results. Each developer's contribution is packaged as a SearchMonkey application, which users can choose to add to their Yahoo! profile to affect the presentation of their search results. Each SearchMonkey application is given access to the Microformats and RDFa embedded within the individual search result page, which it can use to provide enhanced results.

A number of microformat vocabularies as well as RDFa are also directly searchable via Yahoo!, for example, a search term of the form

```
searchmonkeyid:com.yahoo.page.uf.hcard
```

would return all pages that use the hCard microformat. Similar search terms can be used to detect RDFa in a page. Though the detailed semantic data are not yet indexed and thus are not yet searchable directly, the presence of Microformats or RDFa can be used to filter the results.

In early 2009, Yahoo! enabled SearchMonkey support in BOSS, the Build-Your-Own-Search-Service product where developers can directly call into the Yahoo! search API. Using this service, developers can develop vertical-specific search engines that utilize embedded semantic data to provide rich results.

Of course, the desired next step is the ability to alter the results themselves based on the semantics, not just their presentation. At this point, only prototypes of these search engines exist. One hopes that Yahoo!'s SearchMonkey will evolve to provide this full-fledged semantic search.

5.2.2.2 Google

In May 2009, Google also announced their plans to use Microformats and RDFa information embedded in Web pages [22]. At first, this information is used by Google to enrich the "snippets," that is, the information the search result listing provides on each entry. Using the Microformat or the RDFa information embedded in the data, the Google search result is able to provide, for example, price ranges and review references to a result on a specific restaurant. In May 2010, this service was extended to what Google calls "Google Squared," which provides even more information on each search result (images as well as text, the exact origin of specific information, etc). Google has also announced, in September 2009, that they would also index Yahoo!'s SearchMonkey RDFa for video (http://developer.search.yahoo.com/help/objects/video, a simple RDFa-based annotation of video embedded in XHTML).

All these are of course the early steps, but it can be expected that embedded semantic data will be used more extensively both by Yahoo! and by Google (and probably by other search engines) in future.

5.2.3 Facebook's Open Graph Protocol

In May 2010 Facebook announced their Open Graph Protocol [23, 24]. The goal of this protocol is to enable any Web page to become a rich object (i.e., a node) in a social graph. A typical usage of this protocol is to add a Facebook "like" button on a page which enables users to make connections to this page and share content back to their friends.

For a page to be part of the graph, it has to contain some metadata. These metadata are added to the page using RDFa [24]. At the time of finalizing this manuscript the RDFa statements appear only in the header of the page and use a few simple terms only, but it is probable that this mechanism will evolve toward more complex vocabularies and usage patterns. A few days after the publication of the protocol a number of sites (e.g., Rotten Tomatoes, NHL, Microsoft, a number of online journals and magazines like the *TIME*) have begun publishing their metadata in RDFa. Some of these sites (e.g., Rotten Tomatoes) have immediately gone beyond the simple Open Graph vocabulary and have added more complex metadata to their pages. In other words, the introduction of this protocol has dramatically increased the information available, if necessary, in RDF thanks to RDFa.

It is interesting to see that, as a result of Yahoo!'s and Google's usage of embedded semantics and, more recently, due to Facebook's Open Graph Protocol, a number of sites have begun to automatically include Microformats or RDFa as part of their Web pages. Typical examples for RDFa usage are the individual slide pages on SlideShare (http://www.slideshare.net/), the publication of the *London Gazette* (http://www.london-gazette.co.uk/), or the product pages of TESCO (http://www.clothingattesco.com) or those of BestBuy (http://www.bestbuy.com). Google's map service (http://maps.google.com) uses the hCard microformat for contact information, which is also used by Twitter (http://www.twitter.com) alongside with some other Microformats.

5.2.4 Browser Extensions

An important application of HTML-embedded metadata such as Microformats and RDFa is the resulting ability to augment the Web-browsing user experience. In particular, semantic data expressed in a Web page can be used by a semantically aware Web browser to connect the current data to past public data seen on other pages or to the user's private data store. These connections may range from simple "related information" linking to complete archival, private search, and reasoning on the stored information.

5.2.4.1 BlueOrganizer

Adaptive Blue (http://www.adaptiveblue.com/) provides a browser add-on. When the add-on encounters a Web page describing products for purchase using Microformats, it connects this content to possible locations where these products can be purchased. For example, a book reviewed on a blog can be automatically linked to its purchase page on Barnes & Noble, Amazon, and others, using the right-click triggered contextual menu. Rather than letting the publisher of the data decide which specific store to link to, the viewer can decide to use their store of choice for the product in question.

5.2.4.2 Operator

Operator (https://addons.mozilla.org/firefox/4106/) is a browser add-on that detects Microformats and RDFa and displays a number of data-dependent menu items that allow the user to take quick actions based on the embedded semantic data. A user is able to save an hCard to his/her address book and an hCalendar event to his/her calendar. A user may also look up where to purchase a song marked up using audio RDFa.

Where BlueOrganizer provides a fixed set of actions which they control, Operator is extensible. Developers can write data-dependent actions known as Operator Scripts, and users can choose to install the Operator Scripts they choose.

5.2.4.3 Fuzz

Fuzz (http://rdfa.digitalbazaar.com/fuzz/trac/) is a browser add-on that detects and parses RDFa using `librdfa`, a high-speed cross-platform RDFa parser. By default, Fuzz displays Friend-Of-A-Friend (FOAF) [14] information. In general, it can be extended to display any vocabulary, or to take any action based on this vocabulary. Digital Bazaar, the creators of Fuzz, are developing an advanced browser-based music-exchange application built on Fuzz.

5.2.5 Creative Commons and the CC Network

A particularly interesting use case for embedded semantic data in (X)HTML is that of loosely coupled mash-ups – structured data on one web page can be dynamically connected to another application, either on the server side or on the browser side. Applications can connect opportunistically; by publishing a particular vocabulary, one website can automatically be connected to another that seeks to consume that vocabulary. There are no domain-specific APIs – the only connection is the common vocabulary.

Creative Commons has long advocated the use of RDFa to mark up digital works and their many properties, including notably their copyright license [25]. For users who become members of the Creative Commons Network, a detailed XHTML+RDFa fragment is provided for each Web-based work that they wish to claim. This fragment contains a link to the appropriate Creative Commons Deed and a link to their Creative Commons Network profile. Both of these links are typed with RDFa. When a visitor to the work clicks the link to the Creative Commons Deed, the Deed checks its HTTP referrer, parses the RDFa, and displays the appropriate attribution name and URL for the work, as well as a new XHTML+RDFa fragment that can be immediately used, when redistributing or modifying the work, to give credit to the original author.

While the Creative Commons Network plays the role of a digital works registry, other registries can just as easily plug into the Creative Commons Deed, simply by reusing the same RDF vocabulary. By feeding purely off the RDFa, the Creative Commons Deed is a loosely coupled Web application, where any publisher who wishes to connect can do so with ease (❯ *Fig. 5.3*).

Under the following conditions:

Attribution — You must attribute this work to **Ben Adida** (with link).

Attribute this work:

`<div xmlns:cc="http://creativecommons.org/ns#" about="http://ben`

�■ **Fig. 5.3**

A portion of the Creative Commons Deed, which displays the attribution information because it has parsed the RDFa from its referrer URL and determined to whom credit should be given

5.2.6 Separating User Interface from Raw Content

Another fascinating use case of embedded semantic data is the ability for Web publishers to compartmentalize the development of raw content from advanced visualization. Talis and the University of Plymouth demonstrated just how to do this by developing an advanced JavaScript user interface that feeds off RDFa markup in a Web page [26]. In their own words:

▶ The interface to build or edit lists uses a WYSIWYG metaphor implemented in Javascript operating over RDFa markup, allowing the user to drag and drop resources and edit data quickly, without the need to round trip back to the server on completion of each operation. The user's actions of moving, adding, grouping or editing resources directly manipulate the RDFa model within the page. When the user has finished editing, they hit a save button which serialises the RDFa model in the page into an RDF/XML model which is submitted back to the server.

Thus, the markup with its embedded semantics feeds both the visual interface and the back-end data store.

5.2.7 Decentralized Data Distribution Using UK Government Websites

The Central Office of Information is responsible for charting the direction of many of the UK government's websites, and a recent project involved creating a centralized location where the public can search job vacancies that are available in the public sector.

Rather than trying to tackle the enormous task of getting each department to enter their vacancy information into a centralized database, the COI instead took a decentralized approach, allowing each department to retain their existing databases and content management systems; the only change required was that each website should now include RDFa in their pages, in order to describe the jobs available. This RDFa could then be consumed by a central server.

Using RDFa in this way ensures that departments of any size can have their vacancies included in the central store, regardless of whether they have an elaborate CMS, or post

jobs by manually creating HTML. But in addition it creates the possibility that other services can also read this information and use it in a more targeted way (❯ *Fig. 5.4*).

This ability for third-parties to also consume the information is particularly important in another UK government project which uses RDFa, that of consultations.

Consultations are a process where a government department issues one or more documents and then invites interested parties to provide feedback. Since any department could issue a consultation on their website, then it can be difficult for the public to find them, unless they know what they are looking for. Since consultation is an essential part of a modern democracy, a number of sites have sprung up that "screen-scrape" the information, but there is always a problem of accuracy.

With RDFa though, each department can mark up their consultations, and, as with vacancies, make this information available to outside services. And since, as shown in an earlier section, search engines such as Google and Yahoo! are starting to index RDFa, a virtuous circle is formed as information becomes even easier for the public to find and make use of.

5.2.8 Publication of Vocabularies and Datasets

Large vocabularies or RDF Data, in RDF(S), OWL, or SKOS are being published on the Web. One of the problems users face is that, in some cases, the size of the vocabulary as a whole is relatively large, which makes it difficult to consult individual terms online. Although these vocabularies are usually downloadable as one or more files encoded in, say,
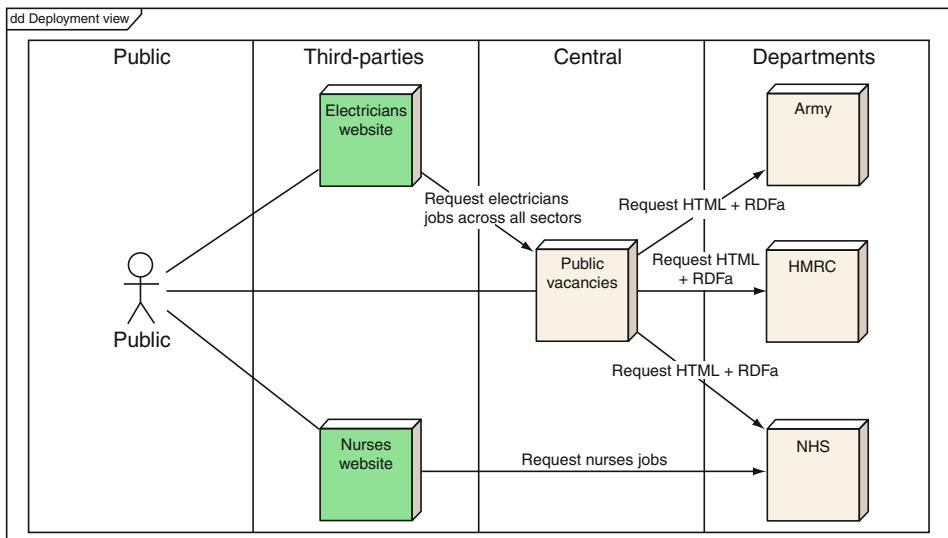
❑ **Fig. 5.4**
RDFa can be used to publish from departments to central servers, and from there to third-party websites

RDF/XML, what the user often wants is simply to get information on a single individual term without the obligation of storing a large file on his or her local disk.

As an example, the Library of Congress (LoC) of the USA has recently made its Authorities and Vocabularies available in RDF, using SKOS [27] to organize the terms and concept hierarchies (http://id.loc.gov/authorities/search/, close to 350,000 concepts are published this way). The importance of this publication is that the Semantic Web community has access to a large number of stable URIs for concepts as different as Semantic Web, or Historical Novel. As an example, the URI:

```
http://id.loc.gov/authorities/sh2002000569#concept
```

can be considered as a reliable reference for the abstract concept for "Semantic Web." The existence of such URIs, and the fact that the corresponding concepts are properly organized, is extremely important for Semantic Web applications. For example, the concept of "World Wide Web" can be considered as a broader concept than the "Semantic Web"; this fact is duly represented using the SKOS term `skos:broader` binding the two corresponding URIs in the LoC dataset. Applications may then make use of these relationships in organizing and integrating their own data.

To help end users, the LoC has set up its website in such a way that each concept URI is redirected to a XHTML page, giving some of the details on a specific concept (see, e.g., ❯ *Fig. 5.5*). That is fine for humans. However, to use the RDF/SKOS triples in an application the developer may have to download the whole dataset (about 34 MB of RDF/XML data) and incorporate it in his/her application. This is obviously not ideal; it may lead to versioning issues, for example.

To facilitate this problem, the XHTML pages on the LoC website use RDFa. Indeed, each page on a particular concept contains all the relevant triples encoded in RDFa. This means that an application can, in real time, access those triples through an RDFa-aware tool, instead of maintaining a local copy of the whole dataset; such a tool will then provide triples like:

```
<http://id.loc.gov/authorities/sh2002000569#concept>
   a skos:Concept ;
   dcterms:created "2004-04-07T00:00:00-04:00"^^xsd:dateTime ;
   skos:broader
      <http://id.loc.gov/authorities/sh95000541#concept>, ... ;
   skos:prefLabel "Semantic Web"@en ;
   ...
```

(where the URI ...sh950000541#concept refers to another concept, in this case that of the World Wide Web).

The Library of Congress is not the only vocabulary or dataset published this way. A similar example is the STW Thesaurus for Economics (http://zbw.eu/stw), published by the German National Library of Economics (considered to be the World's largest economics library) [28]. Yet another example is DBpedia (http://www.dbpedia.org). This dataset is, essentially, a periodic RDF dump of much of the data in Wikipedia enriched by additional links and vocabularies, and made available as Linked Open Data
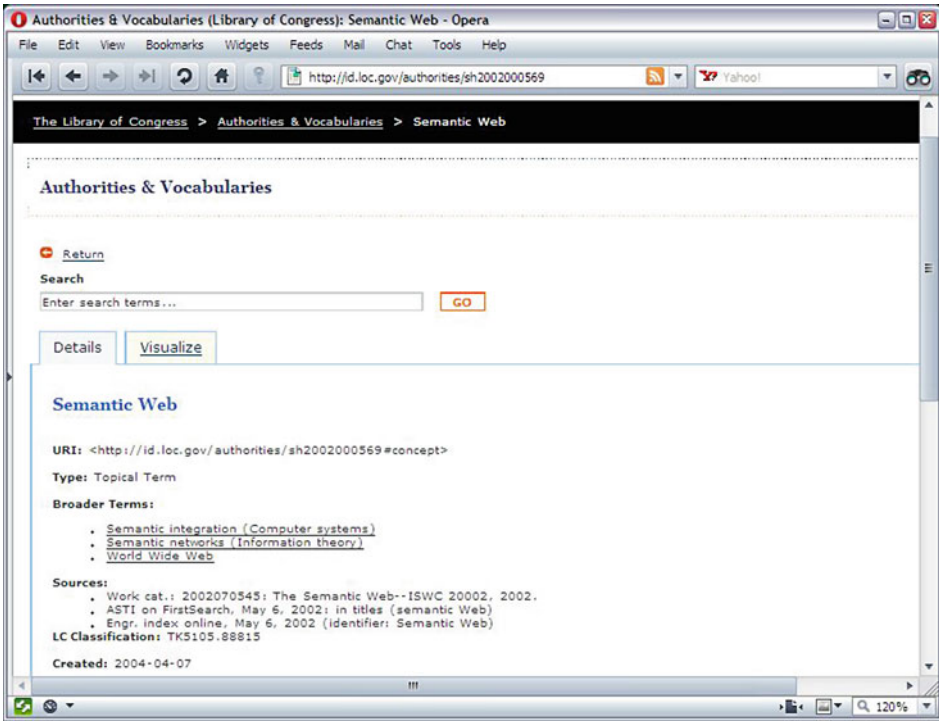
◘ Fig. 5.5
Library of Congress Concept page on Semantic Web

in RDF [29]. While the DBpedia dataset can of course be downloaded, its size makes such a download prohibitive for many applications. However, just like in the LoC or the STW cases, DBpedia URIs can be accessed via a traditional browser, too, with the URIs redirected to XHTML+RDFa pages that contain all the relevant triples for that particular URI. A similar approach is used by dbpedia lite (http://dbpedialite.org) which, in contrast to DBpedia, is not a dump of the Wikipedia dataset but rather a lightweight layer on top of Wikipedias API providing immediate, though for more modest amount of data in RDF.

5.2.9 Self-documenting Vocabulary Specification

One of the well-known problems in software engineering in general is the proper documentation of code. On the Semantic Web, a similar problem occurs when publishing vocabularies: on the one hand, the vocabulary specification must provide a documentation for humans to read and understand; on the other hand, the same vocabulary must have a machine-readable version available in RDF (using, e.g., RDFS, OWL, or SKOS). The traditional approach is to maintain two files side by side; the synchronization between the human-readable and machine-readable format is left to the author(s) of the vocabulary.

RDFa gives the possibility to merge these two. By using XHTML+RDFa, the author can directly encode the RDF triples of the vocabulary definition into the documentation itself. RDFa-aware tools can then extract the RDF content, while humans can read the documentation in their browser. The Biological Taxonomy Vocabulary (http://ontologi. es/biol/ns) or Creative Commons ccREL specification [25] (http://creativecommons.org/ ns#) are good examples.

5.3   Related Resources

The Microformats community maintains its own site (http://microformats.org), where the specifications for all individual Microformats as well as the underlying microformat design principles are made available. Individual microformat specifications often have a reference to example usages on the Web (see, e.g., the "hCard Examples in the Wild" (http:// microformats.org/wiki/hcard-examples-in-wild) page). A number of Microformat tools are also available, both for creation and parsing, and listed on the site (http://microformats. org/code-tools/). Finally, a book on Microformats has also been published [30].

Beyond the specification itself [6], W3C's GRDDL Working Group has also published a GRDDL Primer [13]. The community maintains a Wiki page on GRDDL implementations (http://esw.w3.org/topic/GrddlImplementations) and W3C also provides a general GRDDL Service (http://www.w3.org/2007/08/grddl/) that can be used to extract RDF from a suitably prepared XML file.

Just as in the case of GRDDL, the relevant W3C Working Group has not only published the formal specification of RDFa [7] but it has also provided an RDFa Primer [17]. The RDFa community maintains its own information site (http://rdfa.info), which explains how to use RDFa in a number of specific cases. Tutorials at various conferences are also available online (e.g., [31]). The RDFa info pages include information on various tools for publishing RDFa as well as for parsing and processing.

5.4   Future Issues

The future of the Semantic Web almost certainly relies in large part on its deployment within the existing clickable Web. Microformats and RDFa are two important technologies that will enable this transition to happen. Importantly, both technologies were built to interfere as little as possible with their carrier signal, (X)HTML. As the development effort on HTML5/XHTML5 comes to fruition, it will be possible to include both metadata markup approaches in HTML5 without interfering with HTML's new features.

Note that at the time of finalizing this manuscript, a new RDFa Working Group is already active in defining a slightly updated version of RDFa (referred to as RDFa 1.1). The new developments do not change the nature of RDFa; the goal is rather to simplify the task of RDFa authors in defining easily, for example, commonly used prefixes. (See a recent blog [32] that describes the most important new features.) The HTML Working Group of

W3C has also published a so-called First Public Working Draft on a document outlining the usage of RDFa in HTML5 and XHTML5. While that draft is based on the current version of RDFa, the plan is to harmonize the future evolution of RDFa 1.1 with the final version of HTML5. Finally, the RDFa Working Group has already published a draft for an RDFa API, that is, a programming language interface whereby Web applications may use the RDFa content embedded in a page directly. (See another recent blog [33] giving a short overview of the API features.)

Over time, one can expect new tools to make use of this embedded data to help Web users connect the data they browse more easily. Tools should appear that easily enable the extraction of significant data from one Web page and connect it to other Web pages via their respective structure. One tool that is beginning to provide these features in a prototype environment is Mozilla's Ubiquity (http://labs.mozilla.com/projects/ubiq-uity/), which enables users to dynamically combine various Web Services based on the content they are seeing, for example, map five selected Craigslist apartments on Google Maps (or on Yahoo! Maps if that is the user's preferred option). One can hope to see these types of applications embrace the interoperable structured data stored within Web pages, using Microformats or RDFa.

Finally, another line of development is to use the RDFa approach more widely to non-HTML-related XML dialects. SVG Tiny 1.2 [8] and Yahoo!'s MediaRSS [9] were already mentioned as precursors of this evolution. Another notable example is the inclusion of RDFa-like attributes into the next version of the OpenDocument format (version 1.2) [34], the file format used, for example, by the OpenOffice.org office application. The upcoming RDFa 1.1 version will make this evolution explicit, and will be defined in such a way that any XML dialect will be able to make use of RDFa attributes.

5.5 Cross-References

❯ eGovernment
❯ Semantic Annotation and Retrieval: RDF
❯ Semantic Annotation and Retrieval: Web of Data

References

1. Beckett, D. (ed.): RDF/XML syntax specification (revised), W3C Recommendation. http://www.w3.org/TR/rdf-syntax-grammar/ (2004)

2. Beckett, D., Berners-Lee, T.: Turtle – terse RDF triple language, W3C Team Submission. http://www.w3.org/TeamSubmission/turtle/ (2008)

3. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL query language for RDF, W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/ (2008)

4. Microformats: http://microformats.org

5. Clark, J. (ed.): XSL Transformations (XSLT), W3C Recommendation. http://www.w3.org/TR/xslt (1999)

6. Connolly, D. (ed.): Gleaning Resource Descriptions from Dialects of Languages (GRDDL), W3C Recommendation. http://www.w3.org/TR/grddl/ (2007)

7. Adida, B., Birbeck, M., McCarron, S., Pemberton, S. (eds.): RDFa in XHTML: syntax and processing,

W3C Recommendation. http://www.w3.org/TR/rdfa-syntax (2008)

8. Andersson, O., Berjon, R., Dahlstrm, E., Emmons, A., Ferraiolo, J., Grasso, A., Hardy, V., Hayman, S., Jackson, D., Lilley, C., McCormack, C., Neumann, A., Northway, C., Quint, A., Ramani, N., Schepers, D., Shellshear, A. (eds.): Scalable Vector Graphics (SVG) tiny 1.2 specification, W3C Recommendation. http://www.w3.org/TR/SVGTiny12/ (2008)

9. RSS Advisory Board Media RSS specification version 1.1.2. http://search.yahoo.com/mrss

10. hCalendar – microformats wiki. http://microformats.org/wiki/hcalendar

11. Hayes, P. (ed.): RDF semantics, W3C Recommendation. http://www.w3.org/TR/rdf-mt/ (2004)

12. W3C: Custom RDF dialects – ESW wiki. http://esw.w3.org/topic/CustomRdfDialects

13. Halpin, H., Davis, I.: GRDDL primer, W3C Working Group Note. http://www.w3.org/TR/grddl-primer/ (2007)

14. Brickley, D., Miller, L. (eds.): FOAF vocabulary specification 0.91. http://xmlns.com/foaf/spec/ (2007)

15. Biron, P.V., Malhotra, A. (eds.): XML schema part 2: datatype, 2nd edn., W3C Recommendation. http://www.w3.org/TR/xmlschema-2/ (2004)

16. DCMI Usage Board: DCMI metadata terms. http://dublincore.org/documents/dcmi-terms/ (2008)

17. Adida, B., Birbeck, M. (eds.): RDFa primer, W3C Working Group Note. http://www.w3.org/TR/xhtml-rdfa-primer/ (2008)

18. Gandon, F.: Profile for latest GRDDL transformation for RDFa. http://ns.inria.fr/grddl/rdfa/ (2008)

19. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and consume linked data with Drupal! In: Proceedings of the Eighth International Semantic Web Conference (ISWC 2009), Chantilly. Lecture Notes in Computer Science, vol. 5823, pp. 763–778. Springer, Berlin (2009)

20. SearchMonkey: http://developer.yahoo.com/searchmonkey/

21. Mika, P.: Improving web search using metadata. http://www.w3.org/2001/sw/sweo/public/UseCases/yahoo/ (2008)

22. Goel, K., Guha, R.V., Hansson, O.: Introducing rich snippets. http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html (2009)

23. Open Graph Protocol. http://developers.facebook.com/docs/opengraph

24. Open Graph Protocol. http://opengraphprotocol.org/

25. Abelson, H., Adida, B., Linksvayer, M., Yergler, N. (eds.): ccREL: the creative commons rights expression language, W3C Member Submission. http://www.w3.org/Submission/ccREL/ (2008)

26. Clarke, C., Greig, F.: A linked open data resource list management tool for undergraduate students. http://www.w3.org/2001/sw/sweo/public/UseCases/Talis/ (2009)

27. Miles, A., Bechhofere, S. (eds.): SKOS simple knowledge organization system reference, W3C Recommendation. http://www.w3.org/TR/skos-reference (2009)

28. Borst, T., Neubert, J.: Publishing STW thesaurus for economics as linked open data. http://www.w3.org/2001/sw/sweo/public/UseCases/ZBW/ (2009)

29. Bizer, C., Lehmann, J., Kobilarov, G., Sren, A., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia – a crystallization point for the web of data. J. Web Semant. **7**(3), 154–165 (2009)

30. Allsopp, J.: Microformats, Empowering Your Markup for Web 2.0. Springer, New York (2007)

31. Hausenblas, M., Herman, I., Adida, B.: RDFa – bridging the web of documents and the web of data (tutorial). In: Proceedings of the Seventh International Semantic Web Conference (ISWC 2008), Karlsruhe. Lecture Notes in Computer Science, vol. 5318. Springer, Berlin. http://www.w3.org/2008/Talks/1026-ISCW-RDFa/RDFa-ISWC08.html (2008)

32. Herman, I.: RDFa 1.1 drafts. http://ivan-herman.name/2010/04/22/rdfa-1-1-drafts/

33. Herman, I.: RDFa API draft. http://ivan-herman.name/2010/06/08/rdfa-api-draft/

34. OASIS: OASIS open document format for office applications (OpenDocument) TC. http://www.oasis-open.org/committees/tc home.php?wg abbrev=office