SEMANTIC MARKUP REPORT

MICROFORMATS, RDFA, GRDDL, MICRODATA AND OGP

VESTLANDSFORSKING

Norwegian Centres of Expertise
NCE Tourism Fjord Norway

Contents

Foreword **3**

PART 1

1. Introduction 5
2. Overview of Different Standards 10
2.1 Microformat 10
2.1.1 Microformats and RDF 11
2.2 RDFa 12
2.2.1 RDFa and RDF 12
2.3 GRDDL 13
2.4 Microdata 13
3. Examples 15
3.1 Microformat 15
3.2 RDFa 21
3.3 GRDDL 29
3.3.1 Using GRDDL with Microformat 29
3.3.2 Using GRDDL with RDFa 30
3.4 Microdata 30
3.4.1 Using Microdata with Microformat 33
3.4.2 Common Microdata Formats 34
4. Overview of Facebook's Vocabulary 35
4.1 Open Graph Protocol (OGP) 35
4.1.1 OGP and RDF 35
4.2 Using Open Graph Protocol 36
5. Implementation 39
5.1 Semantic Markup in *Drupal* 39
5.2 Open Graph Protocol in *Drupal* 40
5.3 Semantic Markup and *EPiServer* 40

PART 2

6. **Semantic Support to Existing Pages** 43
6.1 Advantage of Existing Semantic Add Tools 43
6.2 Sample Vocabularies 43
6.3 Examples 46
6.3.1 Dickens Restaurant 46
6.3.2 Tourist Information 48
6.3.3 Concert with Dyvekes Viseklubb 51
6.3.4 Augustin Hotel 55
7. Summary 58
References 60

Foreword

This report has been prepared by Vestlandsforsking as a part of NCE Tourism Fjord Norway initiatives towards the use of semantic annotations in FjordNett web pages. The report addresses semantic markup technologies that have emerged over the last few years to bridge the vast, existing ‘‘clickable’’ Web and the Semantic Web. These markup 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 report, major approaches are explained, exploring their strengths and weaknesses, providing examples, and touching on future considerations for FjordNett.

This work was undertaken with the financial assistance of the NCE Tourism Fjord Norway. We would like to thank Marcel Niederhauser and NCE Tourism for suggesting this project. We hope that this report will be a base for NCE Tourism in realising semantically annotated Tourism websites.

This report is written by Rajendra Akerkar (Vestlandsforsking).

PART 1

1. Introduction

The tourism domain can especially benefit from sophisticated e-Commerce solutions and Semantic Web technology, due to the significant heterogeneity of the market and information sources, and due to the high volume in online transactions. The Semantic Web aims at making the wealth of information that is available on the Web accessible to more precise search and automated information extraction and processing, based on a machine-readable representation of meaning in the form of ontologies (shared vocabulary).

The core components of Semantic Web technology are

1. XML as a generic serialization syntax with mature tool support,
2. Resource Description Framework (RDF) as a data model for the representation of Semantic Networks in a distributed fashion,
3. Ontology languages like RDF-S, OWL, and WSML, for the representation of a domain of discourse,
4. Vocabularies, like e.g. WordNet, Cyc, TOVE, Dublin Core, FOAF, or Harmonise, and
5. Data, which can be regarded part of the vocabulary or not, depending on the respective research community.

Semantic technologies comprise several technological standards where Semantic Web (a W3C standard) and Topic Maps (ISO standard) are the most used and best known. Semantic Web is the leading standard worldwide but in Norway Topic Maps has also gotten a strong hold and is used for instance as a semantic enhancement of VisitNorway.com. This report will only focus on **semantic markup** technologies, which means different ways of including semantic information in the source code (the (X)HTML code). The process of including semantic information (= metadata) is called **annotation**, and means adding machine-readable information to existing content. This way the technology can support a multiplicity of applications, e-g. more precise discovery or adaptation of content. In Tim Berners-Lee's original vision, the entities and relationships between them would be described using RDF. RDF uses this abstract model to decompose information/knowledge into small pieces, with some simple rules about the semantics (meaning) of each one of these pieces. The goal is to provide a general method that is simple and flexible enough to express any fact, yet structured enough that computer applications can operate with the expressed knowledge.

This abstract model has the following key components:

- statement (formally called as a *triple*)
- subject and object resources
- predicate

Therefore, an RDF statement must have the following format:

```
subject  predicate  object
```

where the `subject` and `object` are names for two things in the world, with the `predicate` being the name of a relation (this relation is also sometimes called a property) that connects these two things. The idea behind RDF is to give us a simple way to make statements about things on the web and have machines understand us. Let's look at an example.

Let's say we want to express that the website at http://www.visitbergen.com was created by Bergen Tourist Board. In this case we have the following 3 things that comprise our assertion:

The **subject**: http://www.visitbergen.com

The **predicate** or **property**: creator

The **object** or value: Bergen Tourist Board

Now to put this into RDF syntax, we do the following:

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

         xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="http://www.visitbergen.com">

        <dc:creator>Bergen Tourist Board</dc:creator>

  </rdf:Description>

</rdf:RDF>
```

Here, `<rdf:RDF>` is the root element of an RDF document. It defines the XML document to be an RDF document. It also contains a reference to the RDF namespace. The `<rdf:Description>` element contains elements that describe the resource (subject). `<dc:creator>` is an element from the Dublin Core vocabulary that represents the creator of a given document.

There are three main uses of RDF

- to share simple factual data directly in the Web
- as metadata, to describe other useful information
- semantic annotations

Tagging, is about attaching names, attributes, comments, descriptions to a document or to a selected part in a text. It provides additional information (metadata) about an existing piece of data.

Compared to tagging, which speeds up searching and helps you find relevant and precise information, ***Semantic Annotation*** (Markup) goes one level deeper:

- It enriches the unstructured or semi-structured data with a context that is further linked to the structured knowledge of a domain.
- It allows results that are not explicitly related to the original search.

Therefore, if *tagging* is about promptly finding the most relevant result, *semantic annotation* adds diversity and richness to the process.

Semantic annotation is essentially a meaningful way to describe the structure and appearance of a particular document. Semantic Annotation helps to bridge the ambiguity of the natural language when expressing notions and their computational representation in a formal language. By telling a computer how data items are related and how these relations can be evaluated automatically, it becomes possible to process complex filter and search operations.

To finish the markup process, we have to first create a collection of RDF statements to describe the meaning of a Web document, then put them into a separate file, and finally, we have to somehow link the original Web document to this RDF file. A simpler way of doing all these operations is provided in this document

A simpler approach called **Microformats** was developed by Tantek Celik, Chris Messina and others [microformats.org/]. Unlike RDF, Microformats rely on existing (X)HTML standards and leverage CSS classes to markup the content. Critically, Microformats don't add any additional information to the page, but just annotate the data that is already on the page.

Microformats enjoyed support and wider adoption because of their relative simplicity and focus on marking up the existing content. But there are still issues. First, the number of supported entities is limited, the focus has been on marking organizations, people and events, and then reviews, but there is no way to markup, for example, a movie or a book or a song. Second, Microformats are somewhat cryptic and hard to read. There is cleverness involved in figuring out how to do the markup, which isn't necessarily a good thing.

In 2005, inspired by Microformats, Ian Davis, now CTO of Talis, developed eRDF -- a syntax within HTML for expressing a simplified version of RDF. His approach combined the canonical concepts of RDF and the idea from Microformats that the data is already on the page. Interestingly an iteration of Ian's work, called RDFa, has been adopted as a W3C standard. All the signs point in the direction of RDFa being the solution of choice for describing entities inside HTML pages.

The microformats or RDFa constructs can be *directly* embedded into XHTML to convey the meaning of the document itself, instead of collecting them into separated documents.

Until recently, despite the progress in the markups, adoption was hindered by the fact that publishers lacked the incentive to annotate the pages. What is the point if there are no applications that can take advantage of it? Luckily, in 2009 both Yahoo and Google put their muscle behind marking up pages. First Yahoo developed an elegant search application called *SearchMonkey*.

SearchMonkey is an open platform for using structured data to build more useful and relevant search results.

This app encouraged and enabled sites to take control over how Yahoo's search engine presented the results. The solution was based on both markup on the page and a developer plugin, which gave the publishers control over presenting the results to the user. Later, Google announced **rich snippets**. This supported both Microformats and RDFa markup and enabled webmasters to control how their search results are presented.

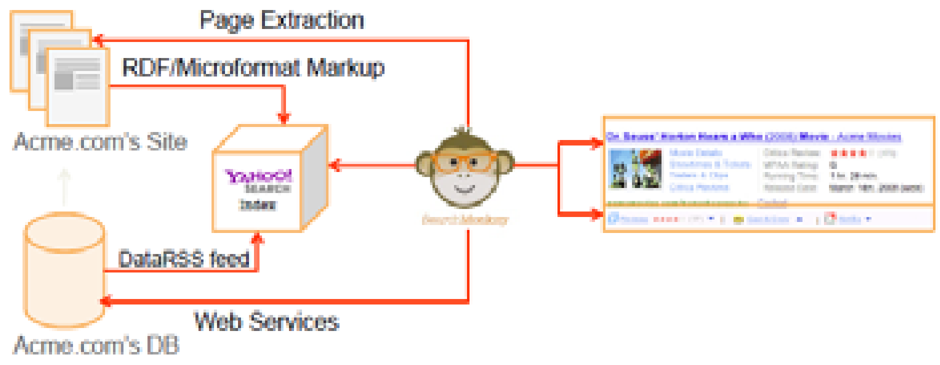
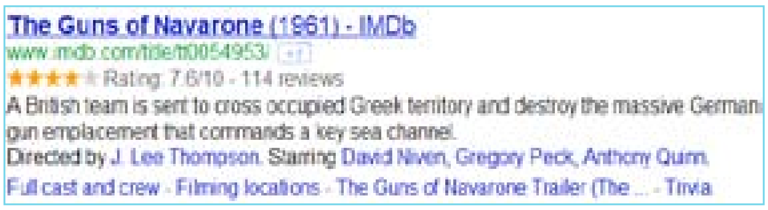


Figure 1.1: How *SearchMonkey* works

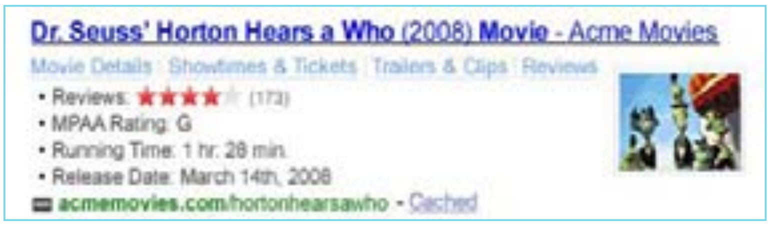
The **main advantage** of the Semantic Markup is improved search, which pulls in information from across the internet and surfaces it in a way that is useful to users. Examples of this are already being seen in major search engines like Bing, Google, and Yahoo! using RDFa to surface information in searches that provide more information than just a result.

For two such examples, we did searches for two movies on the Google and Yahoo! search engines, with returning rich results.



The Guns of Navarone (1961) - IMDb
www.imdb.com/title/tt0054953/
★★★★ Rating: 7.6/10 - 114 reviews
A British team is sent to cross occupied Greek territory and destroy the massive German gun emplacement that commands a key sea channel.
Directed by J. Lee Thompson. Starring David Niven, Gregory Peck, Anthony Quinn.
Full cast and crew - Filming locations - The Guns of Navarone Trailer (The ... - Trivia

Figure 1.2: Example of Google's Rich Snippets when searching for The Guns of Navarone



Dr. Seuss' Horton Hears a Who (2008) Movie - Acme Movies
Movie Details | Showtimes & Tickets | Trailers & Clips | Reviews
- Reviews: ★★★★ (173)
- MPAA Rating: G
- Running Time: 1 hr. 28 min.
- Release Date: March 14th, 2008

acmemovies.com/hortonhearsawho - Cached

Figure 1.3: Example of Yahoo!'s SearchMonkey application

The idea behind Rich Snippets is quite straightforward: it is created by using the structured data embedded in Web pages, and the structured data are added by Web page authors. More specifically, the crawler still works as usual, i.e., traveling from page to page and downloading the page content along its way. However, the indexer's work has changed quite a bit: when indexing a given page, it also looks for the markup formats Google supports. Once some embedded markups are found, they will be collected and will be used to generate the Rich Snippets.

2 Overview of Different Standards

2.1 Microformats

To put it simple, microformats are a way to embed specific semantic data into the HTML content that we have today, so when a given application accesses this content, it will be able to tell what this content is about.

We are familiar with HTML pages that represent people, so let us start from here. Let us say we would like to use microformats to add some semantic data about people. To do so, we need the so-called `hCard` microformat, which offers a group of constructs you can use to mark up the content:

- a root class called `vcard`;
- a collection of properties, such as `fn` (formatted name) and `n` (name), and quite a
- few others.

`hCard` microformat can be used to mark up the page content where a person is described. In fact, `hCard` microformat not only is used for people, but can also be used to mark up the content about companies, organizations and places, as we will see in the next section.

Now, what if we would like to mark up some other content? For instance, some event described in a Web document. In this case, we will need to use the `hCalendar` microformat, which also provides a group of constructs we can use to mark up the related content:

- a root class called `vcalendar`;
- a collection of properties, such as `dtstart`, summary, location, and quite a few others.

By the same token, if we would like to mark up a page content that contains a cooking recipe in restaurant, we then need to use the `hRecipe` microformat.

We can define microformats as follows:

Microformats are a collection of individual microformats, with each one of them representing a specific domain (such as person, event, location) that can be described by a Web content page. Each one of these microformats provides a way of adding semantic markups to these Web pages, so that the added information can be extracted and processed by software applications.

With this definition in mind, it is understandable that the microformats collection is always growing: there are existing microformats that cover a number of domains, and for the domains that have not been covered yet, new microformats are created to cover them.

For example, `hCard` microformat and `hCalendar` microformat are stable micro- formats; `hResume` microformat and `hRecipe` microformat are still in draft states. In fact, there is a microformats community that is actively working on new micro- formats.

Finally, note that microformats are not a W3C standard or recommendation. They are offered by an open community and are open standards originally licensed under Creative Commons Attribution. They have been placed into the public domain since 29 December 2007.

2.1.1 Microformats and RDF

In this section, we will first summarize the benefits offered by microformats, and more importantly, we will also take a look at the relationship between microformats and RDF.

What Is Special About Microformats?

Microformats do not require any new standards; instead, they leverage existing standards. For example, microformats reuse HTML tags as much as possible, since almost all the HTML tags allow class attributes to be used.

Second, the learning curve is minimum for content publishers. They continue to mark up their Web documents as they normally would. The only difference is that they are now invited to make their documents more semantically rich by using class attributes with standardized properties values, such as those from `hCard` microformat.

Third, the added semantic markup has no impact on the document's presentation, if it is done right.

Lastly, and perhaps the most important one, is the fact that this small change in the markup process does bring a significant change to the whole Web world. The added semantic richness can be utilized by different applications, since applications can start to understand at least part of the document on the Web now.

Microformats and RDF

Obviously, the primary advantage microformats offer over RDF is the fact that we can embed metadata directly in the XHTML documents. This not only reduces the amount of markup we need to write, but also provides one single content page for both human readers and machines. The other advantage of microformats is that microformats have a simple and intuitive syntax.

However, microformats were not designed to cover the same scope as RDF was, and they simply do not work on the same exact level. To be more specific, the following are something offered by RDF, but not by microformats.

- RDF does not depend on pre-defined "formats," and it has the ability to utilize, share, and even create any number of vocabularies.

- With the help from these vocabularies, RDF statements can participate in reasoning process and new facts can be discovered by machines.
- Resources in RDF statements are represented as URIs, allowing a Linked Data Web to be created.
- RDF itself is infinitely extensible and open-ended and hence much more flexible.

2.2 RDFa

RDFa is quite simple to understand: it is just another way to directly add semantic data into XHTML pages. Unlike microformats which reuse the existing class attribute on most HTML tags, RDF provides a set of new attributes that can be used to carry the added markup data. Therefore, in order to use RDFa to embed the markup data within the Web documents, some attribute-level extensions to XHTML have to be made. In fact, this is also the reason for the name: RDF*a* means RDF in HTML *a*ttributes.

Note that unlike microformats, RDFa is a W3C standard. More specifically, it became a W3C standard on 14 October 2008. Based on this standard document, RDFa is officially defined as follows:

RDFa is a specification for attributes to express structured data in any markup language.

Another W3C RDFa document, *RDFa for HTML Authors*, has provided the following definition of RDFa:

RDFa is a thin layer of markup you can add to your web pages that make them understandable for machines as well as people. By adding it, browsers, search engines, and other software can understand more about the pages, and in so doing offer more services or better results for the user.

2.2.1 RDFa and RDF

What Is Special About RDFa?

The benefits offered by microformats are all still true for RDFa, and we can add one more here: RDFa is useful because microformats only exist as a collection of centralized vocabularies. More specifically, what if we want to mark up a Web page about a resource, for which there is no microformat available to use? In that case, RDFa is always a better choice, since you can in fact use any vocabulary for your RDFa markup.

RDFa and RDF

To put it simple, RDFa is just a way of expressing RDF triples inside given XHTML pages.

However, RDFa does make it much easier for people to express semantic information in conjunction with a normal Web page. For instance, while there are many ways to express RDF (such as in serialized XML files that live next to standard Web pages), RDFa helps machines and humans read exactly the same content. This is one of the major motivations for the creation of RDFa.

Having a HTML representation and a separate RDF/XML representation (or N3 and Turtle, etc.) is still a good solution for many cases, where HTTP content negotiation is often used to decide which format should be returned to the client.

2.3 GRDDL

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) is a way (a markup format, to be more precise) that enables users to obtain RDF triples out of XML documents (called *XML dialects*), in particular XHTML documents. The following GRDDL terminologies are important for us to understand GRDDL (pronounced 'griddl'):

- *GRDDL-aware agent*: a software agent that is able to recognize the GRDDL transformations and run these transformations to extract RDF.
- *GRDDL Transformation*: an algorithm for getting RDF from a source document.

GRDDL became a W3C Recommendation on 11 September 2007. In this standard document, GRDDL is defined as the following:

*GRDDL is a mechanism for **G**leaning **R**esource **D**escriptions from **D**ialects of **L**anguages. The GRDDL specification introduces markup based on existing standards for declaring that an XML document includes data compatible with RDF and for linking to algorithms (typically represented in XSLT), for extracting this data from the document.*

2.4 Microdata

Microdata is the most recent competitor on the block, and is a format proposed by the W3C as a part of HTML5, and it appears to be heavily influenced by Microformats, adopting a number of the same label names found in `hCard`, `hCalendar` and other Microformats.

Apart from the semantic elements HTML5 introduces Microdata – the way of annotating web pages with semantic metadata using just DOM attributes, rather than separate XML documents. Microdata annotates the DOM with scoped name/value pairs from custom vocabularies. Anyone can define a microdata vocabulary and start embedding custom properties in their own web pages. Every microdata vocabulary defines a set of named properties.

For example, a Person vocabulary could define properties like name and photo. To include a specific microdata property on your web page, you provide the property name in a specific place. Depending on where you declare the property name, microdata has rules about how to extract the property value. Defining your own microdata vocabulary is easy. First, you need a namespace, which is just a URL. The namespace URL could actually point to a working web page, although that's not strictly required.

What Is Special About Microdata?

Microformat looks very similar to the microformats, but the new `item` and `itemprop` attributes are used instead of `class`. The `subject` attribute can be used to avoid the "common ancestor" problem we have with microformats by simply referring to the item element by id.

Microdata and RDFa

Microdata cannot express two things that RDFa supports: datatypes of literals, and XML literals. As part of the ongoing discussion about how to reconcile RDFa and microdata, Nathan Rixham[1] has put together a suggested Microdata RDFa Merge that brings together parts of microdata and parts of RDFa, creating a completely new set of attributes, but a parsing model that more or less follows microdata's.

[1] http://www.jenitennison.com/blog/node/162

3. Examples

This section is devoted to a closer look at how to use markups discussed earlier in a given web document.

3.1 Microformats

We will focus on `hCard` microformat in this section. `hCard` microformat is considered to be one of the most popular and well-established microformats. We will begin with an overview of `hCard` microformat, followed by some necessary HTML knowledge, and as usual, we will learn `hCard` by examples.

From vCard to hCard Microformat

`hCard` microformat has its root in `vCard` and can be viewed as a `vCard` representation in HTML, hence the letter *h* in `hCard` (HTML `vCard`). It is therefore helpful to have a basic understanding of `vCard`.

Table 3.1 Example properties contained in vCard standard

Property name	Property description	Semantic
`N`	Name	The name of the person, place, or thing associated with the `vCard` object
`FN`	Formatted name	The formatted name string associated with the `vCard` object

`TEL`
`EMAIL`
`URL`
	Telephone E-mail URL	Phone number string for the associated `vCard` object
		E-mail address associated with the `vCard` object
		A URL that can be used to get online information about the `vCard` object

`vCard` is a file format standard that specifies how basic information about a person or an organization should be presented, including name, address, phone numbers, e-mail addresses and URLs. This standard was originally proposed in 1995 by the Versit Consortium, which had Apple, AT&T Technologies, IBM and Siemens as its members. In late 1996, this standard was passed on to the Internet Mail Consortium, and since then it has been used widely in address book applications to facilitate the exchange and backup of contact information.

To this date, this standard has been given quite a few extensions, but its basic idea remains the same: vCard has defined a collection of properties to represent a person or an organization. Table 3.1 shows some of these properties.

Since this standard was formed before the advent of XML, the syntax is just simple text that contains property–value pairs. For example, `vCard` object can be expressed as shown in Example 1.

Example 1 `vCard` object

```
BEGIN:VCARD FN:Marcel Niederhauser N:Niederhauser;Marcel;;;
URL:http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-Niederhauser
END:VCARD
```

Note this `vCard` object has a `BEGIN:VCARD` and `END:VCARD` element, which marks the scope of the object. Inside the object, the `FN` property has a value of Marcel Niederhauser, which is used as the display name. The `N` property represents the structured name, in the order of first, last, middle names, prefixes and suffixes, separated by semicolons. This can be parsed by a given application so as to understand each component in the person's name. Finally, URL is the URL of the Web site that provides more information about the vCard object.

With the understanding about `vCard` standard, it is much easier to understand `hCard` microformat, since it is built directly on the `vCard` standard. More specifically, the properties supported by the `vCard` standard are mapped directly to the properties and sub-properties contained in `hCard` microformat, as shown in Table 3.2.

Table 3.2 Examples of mapping `vCard` properties to `hCard` properties

vCard property	hCard properties and sub-properties
`FN`	`fn`
`N`	`n` with sub-properties: family-name, given-name, additional-name, honorific-prefix, honorific-suffix
EMAIL	email with sub-properties: type, value
URL	url

Note Table 3.2 does not include all the property mappings, and you can find the complete mappings from microformats' official Web site. As a high-level summary, `hCard` properties can be grouped into six categories:

- Personal information properties: these include properties such as `fn`, `n`, `nickname`.
- Address properties: these include properties such as `adr`, with sub-properties such as `street-address`, `region` and `postal-code`.
- Telecommunication properties: these include properties such as `email`, `tel`, and `url`.
- Geographical properties: these include properties such as `geo`, with sub-properties such as `latitude` and `longitude`.
- Organization properties: these include properties such as `logo`, `org`, with sub-properties such as `organization-name` and `organization-unit`.
- Annotation properties: these include properties such as `title`, `note`, and `role`.

With the above mapping in place, the next issue is to represent a `vCard` object (contained within `BEGIN:VCARD` and `END:VCARD`) in `hCard` microformat. To do so, `hCard` microformat uses a root class called `vcard`, and in HTML content, an element with a class name of `vcard` is itself called an `hCard`.

Now, we can look at some examples to understand how exactly we can use `hCard` microformat to mark up some page content.

Using `hCard` Microformat to Mark Up Page Content

Let us start with a very simple example. Suppose that in one Web page, we have some HTML code as shown in Example 2.

Example 2 Example HTML code without `hCard` microformat markup

```
... <!-- other HTML code -->
<div>
<a href="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser/">Marcel Niederhauser</a>
</div>
... <!-- other HTML code →
```

Obviously, for our human eyes, we understand that the above link is pointing to a Web site which describes a person named Marcel Niederhauser. However, any application that sees this code does not really understand that, except for showing a link on the screen as follows:

Marcel Niederhauser

Now let us use `hCard` microformat to add some semantic information to this link. The basic rules when doing markup can be summarized as follows:

- use `vcard` as the class name for the element that needs to be marked up, and this element now becomes a `hCard` object, and

- the properties of an `hCard` object are represented by elements inside the `hCard` object. An element with class name taken from a property name represents the value of that property. If a given property has sub-properties, the values of these sub-properties are represented by elements inside the element for that given property.

Based on these rules, Example 3 shows one possible markup implemented by using `hCard` microformat.

Example 3 `hCard` microformat markup added to Example 2

```
... <!-- other HTML code -->
<div class="vcard">
<div class="fn">Marcel Niederhauser</div>
<div class="n">
<div class="given-name">Marcel</div>
<div class="family-name">Niederhauser</div>
</div>
<div class="url">http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser</div>
</div>
... <!-- other HTML code -->
```

This markup is not hard to follow. For example, the root class has a name given by `vcard`, and the property names are used as class names inside it. And certainly, this simple markup is able to make a lot of difference to an application: any application that understands `hCard` microformat will be able to understand the fact that this is a description of a person, with the last name, first name and URL given.

If you open up Example 3 using a browser, you will see it is a little bit different from the original look-and-feel. Instead of a clickable name, it actually shows the full name, first name, last name and the URL separately. So let us make some changes to our initial markup, without losing the semantics, of course.

First off, a frequently used trick when implementing markup for HTML code comes from the fact that class (also including `rel` and rev attributes) attribute in HTML can actually take a space-separated list of values. Therefore, we can combine `fn` and `n` to reach something as shown in Example 4.

Example 4 An improved version of Example 3

```
... <!-- other HTML code -->
<div class="vcard">
```

```
<div class="n fn">
<div class="given-name">Marcel</div>
<div class="family-name">Niederhauser</div>
</div>
<div class="url">http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser</div>
</div>
... <!-- other HTML code -->
```

This is certainly some improvement: at least we don't have to encode the name twice. However, if you open up Example 4 in a browser, it still does not show the original look. To go back to its original look, at least we need to make use of element `<a>` together with its `href` attribute.

In fact, microformats do not force the content publishers to use specific elements; we can choose any element and use it together with the class attribute. Therefore, Example 5 will be our best choice.

Example 5 Final `hCard` microformat markup for Example 2

```
... <!-- other HTML code -->
<div class="vcard">
<a class="n fn url" href="http://www.fjordnorway.com/no/NCE/KONTAKT-
OSS/Marcel-Niederhauser">
<span class="given-name">Marcel</span>
<span class="family-name">Niederhauser</span>
</a>
</div>
... <!-- other HTML code -->
```

If you open up Example 5 from a Web browser, you get exactly the original look-and-feel. And certainly, any application that understands `hCard` microformat will be able to understand what a human eye can see: this is a link to a Web page that describes a person, whose last name is Niederhauser and first name is Marcel.

Example 6 is another example of using `hCard` microformat. It is more complex and certainly more interesting.

Example 6 A more complex `hCard` microformat markup example

```
<div id="hcard-Marcel-Niederhauser" class="vcard">
```

```
<a  class="n  fn  url"  href="http://www.fjordnorway.com/no/NCE/KONTAKT-
OSS/Marcel-Niederhauser">
<span class="given-name">Marcel</span>
<span class="family-name">Niederhauser</span>
</a>
<div class="org">NCE Tourism-Fjord Norway</div>
<div class="tel">
<span class="type">work</span>
<span class="value">47 955 56 256</span>
</div>
<div class="adr">
<div class="street-address">Lodin Leppsgt. 2b</div>
<span class="locality">Bergen</span>,
<span class="region">NO</span>
<span class="postal-code">5003</span>
<div class="country-name">Norway</div>
</div>
<a class="email" href="mailto:Marcel.Niederhauser@vestforsk.no">
Marcel.Niederhauser@vestforsk.no
</a>

</div>
```

And Example 7 shows the result rendered by a Web browser.

Example 7 Rendering result of Example 6

```
Marcel Niederhauser
NCE Tourism-Fjord Norway
Lodin Leppsgt. 2b, NO 5003 Bergen
Norway
Tel (Work) 47 955 56 256
Marcel.Niederhauser@vestforsk.no
```

3.2 RDFa

The attributes introduced by RDFa have names. For example, property is one such attribute. Obviously, when we make reference to this attribute, we say attribute property. In order to avoid repeating the word attribute too often, attribute property is often written as `@property`. We will write `@`*`attributeName`* to represent one attribute whose name is given by *`attributeName`*.

The following attributes are used by RDFa:

`about`	`resource`
`content`	`rev`
`datatype`	`role`
`href`	`src`
`property`	`typeof`
`rel`	

Let us see what XHTML elements these attributes can be used.

The rule is very simple: you can use these attributes to just about any element. For example, you can use them on `div` element, on `p` element, or even on `h2` (or `h3`, etc.) element. In real practice, there are some elements that are more frequently used with these attributes.

The first such element is the span element. It is a popular choice for RDFa simply because you can insert it anywhere in the body of an XHTML document. Link and meta elements are also popular choices, since you can use them to add RDFa markups to the head element of a HTML document. This is in fact one of the reasons why RDFa is gaining popularity: these elements have been used to add metadata to the head element for years; therefore, any RDFa-aware software can extract useful metadata from them with only minor modifications needed.

The last frequently used element when it comes to add RDFa markup into the content is a linking element. In fact, we can always use `@rel` on a link element to add more information about the relationship, and this information serves as the predicate of a triple stored in that a element.

RDFa: Rules and Examples

In this section we will see how to use RDFa to mark up a given content page, and we will also summarize the related rules when using the RDFa attributes.

RDFa Rules

Any given RDF statement has three components: subject, predicate and object. It turns out that RDFa attributes are closely related to these components:

- Attributes `rel`, `rev` and property are used to represent predicates.
- For attribute `rel`, its subject is the value of about attribute, and its object is the value of `href` attribute.
- For attribute `rev`, its subject and object are reversed compared to `rel:` its subject is the value of `href` attribute, and its object is the value of about attribute.
- For attribute property, its subject is the value of about attribute, and its object is the value of content attribute.

Table 3.3 RDFa attributes as different components of an RDF statement

Object values	Subject attribute	Predicate attribute	Object
Literal strings	about	property	Value of content attribute
Resource (identified by URI)	about	`rel`	Value of `href` attribute

One has to be careful about the object of a given RDF statement: its object can either take a literal string as its value or use another resource (identified by a URI) as its value. How is this taking effect when it comes to RDFa? Table 3.3 summarizes the rules.

Based on Table 3.3, if the object of an RDF statement takes a literal string as its value, this literal string will be the value of content attribute. Furthermore, the subject of that statement is identified by the value of about attribute, and the predicate of that statement is given by the value of property attribute. If the object of an RDF statement takes a resource (identified by a URI) as its value, the URI will be the value of `href` attribute. Furthermore, the subject of that statement is identified by the value of about attribute, and the predicate of that statement is given by the value of `rel` attribute.

Let us see some examples along this line. Assume Marcel has posted an article about the Bergen Tourist Information on Web site. In that post, we have some simple HTML code as shown in Example 8.

Example 8 Some simple HTML code in my article about the Bergen Tourist Information

```
<div>
<h2>This article is about the Bergen Tourist Information and written by
Marcel.</h2>
</div>
```

This can be easily understood by a human reader of the article. First, it says this article is about the Bergen Tourist Information; second, it says the author of this article is Marcel. Now I would like to use RDFa to add some semantic markup, so that machine can see these two facts. One way to do this is shown in Example 9.

Example 9 Use RDFa to mark up the content HTML code in Example 8

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
<p>This article is about <span about="
http://www.fjordnorway.com/no/NCE/Bergen.html" rel="dc:subject" href="
http://dbpedia.org/page/Bergen"/>the Bergen Tourist Information and
written by <span about="http://www.fjordnorway.com/no/NCE/KONTAKT-
OSS/Marcel-Niederhauser/article/Bergen" property="dc:creator"
content="Marcel"/> Marcel.</p>
</div>
```

Recall that dc represents Dublin Core vocabulary namespace. We can pick up the RDFa markup segments from Example 9 and show them in Example 10.

Example 10 RDFa markup text taken from Example 9

```
<span about="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser/article/Bergen.html" rel="dc:subject"
href=" http://dbpedia.org/page/Bergen"/>

<span about="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser/article/Bergen.html" property="dc:creator"
content="Marcel"/>
```

Clearly, in the first span segment, the object is a resource identified by a URI. Therefore, `@rel` and `@href` have to be used as shown in Example 10. Note that http://dbpedia.org/page/Bergen is used as the URI identifying the object resource. This is an URI created by DBpedia project to represent the concept of Bergen. Here we are reusing this URI instead of inventing our own.

On the other hand, in the second span segment, the object is represented by a literal string. Therefore, `@property` and `@content` have to be used.

The last rule we need to discuss here is about attribute *about*. At this point, we understand attribute about is used to represent the subject of the RDF statement. But for a given XHTML content marked up by RDFa, how does an RDFa-aware application exactly identify the subject of the markup? This can be summarized as follows:

- If attribute about is used explicitly, then the value represented by about is the subject.
- If an RDFa-aware application does not find about attribute, it will assume that the about attribute on the nearest ancestor element represents the subject.

- If an RDFa-aware application searches through all the ancestors of the element with RDFa markup information and does not find an about attribute, then the subject is an empty string and will effectively indicate the current document.

These rules about subject are in fact quite intuitive, especially the last one, given the fact that lots of a document's markup information will be typically about the document itself.

RDFa Examples

In this section, we will use examples to show how semantic markup information can be added by using RDFa attributes.

A common usage of RDFa attributes is to add *inline* semantic information. This is in fact the original motivation that led to the creation of RDFa: how to take human- readable Web page content and make it machine readable. Example 9 is a good example of this inline markup. You can compare Example 8 with Example 9; Example 8 is the original page content that is written for human eyes, and Example 9 is what we have after inline RDFa markup. Note that the presentation rendered by any Web browser does not alter at all.

Another example is to mark up the HTML code shown in Example 2. It is a good exercise for us since we have already marked up Example 2 using `hCard` microformats, and using RDFa to mark up the same HTML content shows the difference between the two.

Example 11 shows the RDFa markup of Example 2. It accomplishes the same goal as shown in Example 5. It tells an RDFa-aware application the following fact: this is a link to the home page of a person, whose first name is Marcel and last name is Niederhauser.

Example 11 RDFa markup for the HTML code shown in Example 2

```
... <!-- other HTML code -->
<div xmlns:foaf="http://xmlns.com/foaf/0.1/">
<a about="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser#Marcel" rel="foaf:homepage"
href="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser/">Marcel Niederhauser</a>
<span property="foaf:firstName" content="Marcel"/>
<span property="foaf:lastName" content="Niederhauser"/>
</div>
... <!-- other HTML code -->
```

Again, if you open up the above with a Web browser, you see the same output as given by Example 2. With what we have learned so far, understanding Example 11 should not be difficult at all.

Note that FOAF vocabulary is used for RDFa to mark up the content. For now, just remember FOAF is a vocabulary, with a collection of words that one can use to describe people and their basic information.

This is in fact an important difference between microformats and RDFa. More specifically, when using microformats to mark up a given document, the possible values for the properties are pre-defined. For example, if `hCard` microformat is used, only `hCard` properties and sub-properties can be used in the markup (see Example 5 for example). However, this is not true for RDFa markup: you can in fact use anything as the values for the attributes. For example, Example 11 could have been written as the one shown in Example 12.

Example 12 Another version of Example 11

```
... <!-- other HTML code -->
<div xmlns:Niederhauser="http://www.fjordnorway.com/no/NCE/KONTAKT-
OSS/Marcel-Niederhauser/Niederhauser">
<a about="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser#Marcel" rel="Niederhauser:myHomepage"
href="http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser/">Marcel Niederhauser</a>
<span property="Niederhauser:myFirstName" content="Marcel"/>
<span property="Niederhauser:myLastName" content="Niederhauser"/>
</div>
... <!-- other HTML code -->
```

However, this is not a desirable solution at all. In order for any RDFa-aware application to understand the markup in Example 12, that application has to understand your vocabulary first. And clearly, if all the Web publishers went ahead to invent their own keywords, the world of available keywords would have become quite messy. Therefore, it is always the best choice to use words from a well-recognized vocabulary when it comes to mark up your page. Again, FOAF vocabulary is one such well- accepted vocabulary, and if you use it in your markup (as shown in Example 11) chance is any application that understands RDFa will be able to understand FOAF as well.

In fact, this flexibility of the possible values of RDFa attributes is quite useful for many markup requirements. For example, assume in Marcel's Web site, we have the following HTML snippet as shown in Example 13.

Example 13 HTML code about friend, David

```
... <!-- other HTML code -->
<div>
<p>My friend, David, also likes skiing.</p>
</div>
... <!-- other HTML code -->
```

And I would like to mark up the code in Example 13 so that the machine will understand these facts: first, Marcel has a friend whose name is David; second, David likes skiing.

You can certainly try to use microformats to reach the goal; however, RDFa seems to be quite easy to use, as shown in Example 14.

Example 14 RDFa markup of Example 13

```
... <!-- other HTML code -->
<div xmlns:foaf="http://xmlns.com/foaf/0.1/">
<p>My friend, <span about="http://www.fjordnorway.com/no/NCE/KONTAKT-
OSS/Marcel-Niederhauser#Marcel" rel="foaf:knows"
href="http://www.example.org#david">David
</span>, also likes <span about="http://www.example.org#david "
rel="foaf:interest" href=”http://dbpedia.org/page/Skiing">skiing.
</span></p>
<span about="http://www.example.org#ding" property="foaf:
title"content="Mr."/> <span about="http://www.ex-ample.org#david"
proerty="foaf:lastName" content="David"/>
</div>
... <!-- other HTML code -->
```

Again, note that `http://dbpedia.org/page/Skiing` is used as the URI identifying skiing as a recreational activity. This is also a URI created by DBpedia project. We are reusing this URI since it is always good to reuse existing ones. On the other hand, `http://www.example.org#david` is a URI that we invented to represent Mr. David, since there is no URI for this person yet.

An application which understands RDFa will generate the RDF statements as shown in Example 15 from Example 14 (expressed in Turtle format).

Example 15 RDF statements generated from the RDFa markup in Example 14

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
```

```
<http://www.fjordnorway.com/no/NCE/KONTAKT-OSS/Marcel-
Niederhauser#Marcel>
foaf:knows <http://www.example.org#david>.
<http://www.example.org#david>
foaf:interest <http://dbpedia.org/resource/Skiing>.
<http://www.example.org#david> foaf:title "Mr.".
<http://www.example.org#david> foaf:lastName "David".
```

So far, all the examples we have seen are about inline markup. Sometimes, RDFa semantic markup can also be added about the containing document without explicitly using attribute about. Since this is a quite common use case of RDFa, let us take a look at one such example. Example 16 shows the markup that can be added to the document header.

Example 16 RDFa markup about the containing document

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/">
<head>
<meta property="dc:title" content="Marcel Niederhauser's Homepage"/>
<meta property="dc:creator" content="Marcel Niederhauser"/>
</head>
<body>
<!-- body of the page -->
```

Based on the RDFa rules, when no subject is specified, an RDFa-aware application assumes an empty string as the subject, which represents the document itself.

At this point, we have covered the following RDFa attributes: `about`, `content`, `href`, `property` and `rel`. These are all frequently used attributes, and understanding these can get you quite far already.

The last attribute we would like to discuss here is attribute `typeof`. It is quite important and useful since it presents a case where a blank node is created. Assume on my home page, I have the following HTML code to identify myself as shown in Example 17.

Example 17 HTML code that identifies myself

```
<div>
<p>Marcel Niederhauser</p>
<p>E-mail:<a
href="mailto:marcel@ncetourism.com">marcel@ncetourism.com</a>
</div>
```

We would now like to use RDFa to mark up this part so the machine will understand that this whole div element is about a person, whose name is Marcel Niederhauser and whose e-mail address is marcel@ncetourism.com. Example 18 shows this markup.

Example 18 RDFa markup of the HTML code shown in Example 17

```
<div typeof="foaf:Person" xmlns:foaf="http://xmlns.com/foaf/0.1/">
<p property="foaf:name">Marcel Niederhauser</p>
<p>E-mail: <a rel="foaf:mbox"
href="mailto:marcel@ncetourism.com">marcel@ncetourism.com</a>
</div>
```

Note the usage of attribute `typeof`. More specifically, this RDFa attribute is designed to be used when we need to declare a new data item with a certain type. In this example, this type is the `foaf:Person` type. For now, just understand `foaf:Person` is another keyword from the FOAF vocabulary, and it represents human being as a class called Person.

Now, when `typeof` is used as one attribute on the div element, the whole div element represents a data item whose type is `foaf:Person`. Therefore, once reading this line, any RDFa-aware application will be able to understand this div element is about a person. In addition, `foaf:name` and `foaf:mbox` are used with `@property` and `@rel`, respectively, to accomplish our goal to make the machine understand this information, as you should be familiar by now.

Note we did not specify attribute about like we have done in the earlier examples. So what would be the subject for these properties then? In fact, attribute `typeof` on the enclosing `div` does the trick: it implicitly sets the subject of the properties marked up within that `div`. In other words, the name and e-mail address are associated with a new node of type `foaf:Person`. Obviously, this new node does not have a given URI to represent itself; it is therefore a blank node. Again, this is a trick you will see quite often if you are working with RDFa markup. The last question is if this new node is a blank node, how do we use it when it comes to data aggregation? For example, the markup information here could be quite important; it could be some supplement information about a resource we are interested in. However, without a URI identifying it, how do we relate this information to the correct resource at all?

In this case, the answer is yes. In fact, we can indeed relate this markup information to another resource that exists outside the scope of this document.

3.3 GRDDL

3.3.1 Using GRDDL with Microformats

There are a number of ways to reference GRDDL in a document where micro- formats markup data are added. Referencing GRDDL transformations directly in the head of the HTML document is probably the easiest implementation: only two markup lines are needed.

More specifically, the first thing is to add a profile attribute to the head element to indicate the fact that this document contains GRDDL metadata. Example 19 shows how to do this.

Example 19 Adding profile attribute for GRDDL transformation

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head profile="http://www.w3.org/2003/g/data-view">
<title>Marcel Niederhauser's Homepage</title>
</head>
<body>
<!-- body of the page -->
```

In HTML, profile attribute in head element is used to link a given document to a description of the metadata schema that the document uses. The URI for GRDDL is given by the following,

http://www.w3.org/2003/g/data-view

And by including this URI as shown in Example 19, we declare that the metadata in the markup can be interpreted using GRDDL.

The second step is to add a link element containing the reference to the appropriate transformation. More specifically, recall the fact that microformats is a collection of individual microformats such as `hCard` microformat and `hCalendar` microformat. Therefore, when working with markup data added by using microformats, it is always necessary to name the specific GRDDL transformation.

Let us assume the document in Example 19 contains `hCard` microformat markups. Therefore, the link element has to contain the reference to the specific transformation for converting HTML containing hCard patterns into RDF. This is shown in Example 20.

Example 20 Adding link element for GRDDL transformation (hCard microformat)

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head profile="http://www.w3.org/2003/g/data-view">
```

```
<title>Marcel Niederhauser's Homepage</title>
<link rel="transformation"
href="http://www.w3.org/2006/vcard/hcard2rdf.xsl"/>
</head>
<body>
<!-- body of the page -->
```

These two steps are all there is to it: the profile URI tells a GRDDL-aware application to look for a link element whose `rel` attribute contains the token transformation. Once the agent finds this element, the agent should use the value of `href` attribute on that element to decide how to extract the `hCard` microformat markup data as RDF triples from the enclosing document.

What if `hCalendar` microformat markup has been used in the document? If that is the case, we should use the following transformation as the value of `href` attribute:

`http://www.w3.org/2002/12/cal/glean-hcal.xsl`

3.3.2 Using GRDDL with RDFa

It is basically easy to use GRDDL with RDFa. The first step is still the same, i.e., we need to add a profile attribute to the head element, as shown in Example 19. For the second step, we will have to switch the transformation itself, as shown in Example 21.

Example 21 Adding link element for GRDDL transformation (RDFa)

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head profile="http://www.w3.org/2003/g/data-view">
<title>Marcel Niederhauser's Homepage</title>
<link rel="transformation"
href="http://www.w3.org/2001/sw/grddl-wg/td/RDFa2RDFXML.xsl"/>
</head>
<body>
<!-- body of the page -->
```

3.4 Microdata

Let's say we want to create a microdata vocabulary that describes a person. If we own the data-`vocabulary.org` domain, we'll use the URL `http://datavocabulary.org/Person` as the namespace for microdata vocabulary. That is an easy way to create a globally unique identifier:

pick a URL on a domain that you control. In this vocabulary, we need to define some named properties.

Let's consider three basic properties:

- name (your full name)
- photo (a link to a picture of you)
- `URL` (a link to a site associated with you, like a weblog or a Google profile)

Some of these properties are `URL`s, others are plain text. Each of them lends itself to a natural form of markup, even before you start thinking about microdata or vocabularies. Imagine that you have a profile page or an 'About' page. Your name is probably marked up as a heading, like an `<h1>` element. Your photo is probably an `<img>` element, since you want people to see it. And any `URL`s associated your profile are probably already marked up as hyperlinks, because you want people to be able to click them. For the sake of discussion, let's say your entire profile is also wrapped in a `<section>` element to separate it from the rest of the page content.

Thus:

```
<section        itemscope        itemtype=        "http://data-
vocabulary.org/Person">

 <div itemprop="title" class="title">      Project Manager

  </div>

  <div itemprop="name" class="name">

      Marcel Niederhauser

  </div>

</section>
```

The major advantage of Microdata is its interoperability, i.e. any RDF representation of an ontology can be mapped to HTML5 microdata.

A complete example of using microdata is given in the following lines:

```
<!DOCTYPE html>

<html>
  <head>
    <title>Microdata example</title>

    <script>
```

```
          document.createElement('article');
          document.createElement('section');
          document.createElement('aside');
          document.createElement('hgroup');
          document.createElement('nav');
          document.createElement('header');
          document.createElement('footer');
          document.createElement('figure');
          document.createElement('figcaption');
        </script>

        <style>
          header, footer, section, article, nav, aside, hgroup,
figure, figcaption, video {
            display: block;
          }
        </style>
  </head>
  <body>

  <h1>Me: defined in microdata</h1>

  <article itemscope itemtype="http://example.org/biography">
    <h2 itemprop="name"> Tim Berners-Lee</h2>
    <p><img itemprop="image" src="
http://www.w3.org/Press/Stock/Berners-Lee/2001-europaeum-
eighth.jpg" alt="Photo of Tim - this is me"></p>
    <ul>
      <li>Nationality: <span
itemprop="nationality">British</span></li>
      <li>Age: <span itemprop="age"></span></li>
      <li>Date of birth: <time itemprop="birthday" datetime="1955-
06-08"> 8 June 1955</time></li>
      <li>Hair colour: <span itemprop="colour">Brown</span></li>
      <li>
          <div itemscope itemprop="organisation"
itemtype="http://example.org/organisation" itemref="members">
            <h3>My organisation</h3>
            <ul>
              <li>Name: <span itemprop="name"> World Wide Web 
consortium</span></li>
              <li>Organisation: <span 
itemprop="style">International Standard  for the World Wide Web 
</span></li>
              <li>Members: <span 
itemprop="organisationsize">319</span></li>
            </ul>
          </div>
        </li>
    </ul>
  </article>
  <h3>My organisation</h3>
  <ul id="management team">
      <li itemprop=" management team"> Jeff Jaffe</li>
      <li itemprop=" management team"> J. Alan Bird</li>
      <li itemprop=" management team"> Ian Jacobs</li>
      <li itemprop=" management team"> Ted Guild</li>
      <li itemprop=" management team"> Ralph Swick</li>
    </ul>

    <script type="text/javascript">
    var biography = 
document.getItems("http://example.org/biography")[0];
    alert('Hello ' + biography.properties['name'][0].textContent + 
'!');
    </script>

  </body>
</html>
```

3.4.1 Using Microdata with Microformat

Microformats and microdata can gladly exist together. For example, utilizing both the microformats class values and the microdata properties:

```
<dl itemscope itemtype="http://data-vocabulary.org/Person" class="vcard">
<dt itemprop="name" class="fn"><a href="http://www.ncetourism.com/" itemprop="url" class="url">Marcel Niederhauser</a></dt>
<dd itemprop="title" class="title">Project Manager</dd>
<dd itemprop="address" itemscope itemtype="http://data-vocabulary.org/Address" class="adr"><span itemprop="locality" class="locality">Bergen</span>, <abbr title="Norway" itemprop="region" class="region">NORWAY</abbr> <span itemprop="postal-code" class="postal-code">5003</span></dd>
</dl>
```

3.4.2 Common Microdata Formats

One can define any `itemtype` s/he want for consumption by his/her organization's internal automated bots, but the value of microdata comes from multiple organizations agreeing to use the same vocabulary. In this case, the most immediate benefit for web developers is to use the microdata formats supported by Google's crawlers to feed better, richer data to Google. The clear benefit is the better Google understands your organization, the richer Google's search results will be for your site.

Google's currently supported microdata formats can be found at http://www.google.com/support/webmasters/bin/topic.py?topic=21997.

- *Breadcrumbs*: Defines where a given page is in the overall structure of a site.
- *Businesses and Organizations*: Defines corporate structure and related contact information.
- *Events*: Calendar data.
- *Product Information*: Defines product catalog data.
- *People*: Information about people including contact information.
- *Recipes*: Defines a cooking recipe, nutrition values, ingredients and instructions.
- *Reviews and ratings*: A common interchange format describing reviews and ratings.

4. Overview of Facebook's Vocabulary

4.1 Open Graph Protocol (OGP)

The Open Graph Protocol (OGP) is the markup announced by Facebook that defines several essential attributes -- type, title, URL, image and description. Probably OGP is the most used general ontology on the Web as of today. The protocol comes with a reasonably rich taxonomy of types, supporting entertainment, news, location, articles and general web pages. Facebook hopes that publishers will use the protocol to describe the entities on pages. When users press the *LIKE* button, Facebook will get not just a link, but a specific object of the specific type. OGP redefines vocabulary terms which have been around for many years.

```
og:image        -> foaf:depiction
og:latitude     -> geo:lat
og:postal-code  -> vcard:postal-code
og:email        -> foaf:mbox
og:phone_number -> foaf:phone
```

The OGP enables a web site to display rich content in a social graph.

This is a very simple but powerful concept, one which sites like Facebook have pioneered and leveraged extensively.

In order for a web site to become part of a social graph, you must add a set of metadata to the page. The protocol implemented relies on RDFa. RDFa simply adds a set of extra attributes to the XHTML standard in order to support the extra meta needed for OGP.

- og:title : The title of your object as it should appear within the graph.
- og:type : The type of your object, e.g., "movie". Depending on the type you specify, other properties may also be required.
- og:image : An image URL which should represent your object within the graph.
- og:url : The canonical URL of your object that will be used as its permanent ID in the graph.

With this basic information, a web site can be mapped in a social graph. In addition, there are other types of metadata you can add including: location, audio, video, etc.

4.1.1 OGP and RDF

The Open Graph protocol is built a top of existing Semantic Web standards like RDF and RDFa, (the same standards which have been integrated into the Content Management System *Drupal* in its version 7). Facebook is joining Yahoo SearchMonkey and Google Rich Snippets which now all consume RDFa. Although it has been designed and created by Facebook, OGP can be used by anyone, Facebook being the first consuming this data produced by the sites having the right OGP markup. In fact, there is little information about Facebook on the main OGP documentation page, they even refer the reader to Facebook documentation as "their documentation", keeping OGP as generic as possible. Any web application is free to markup their webpages with the Open Graph protocol markup, and any web application is free to consume this data like Facebook does today - in essence, it's no different than tackling the Semantic Web chicken an egg issue, making the data available as machine readable format (RDFa in this particular case) so that other peers can consume it.

Facebook's OGP is probably the most used semantic markup on the Web today. 10-15% of all FB 'like' are powered by OGP, and that is 10-15% of a very big number. The reason for FB to promote OGP is to enable 'labeled links' and eventually there is where the money is made in terms of tailored advertisings.

What Is Special About OGP?

OGP is a method of telling Facebook more about what your web page is about so that it can be better represented on Facebook. This will improve the matching of users to your page when they search and it will vastly improve the quality of the content that is shown on a user's wall when they like your page.

It involves someone (your web company or web administrator – or even yourself if you have access to your web page such that you can edit your meta data) inserting Facebook specific descriptive meta-tags into the head of your pages. Tags such as "type" allow you to define if the page is about say a hostel or a wine tour. The "`og:image`" tag allows you to specify an image URL such that when someone is liking your page that image will be selected as a thumbnail. The relationship between the information you provide to Facebook through Open Graph protocol and the information that is subsequently used when a person likes you page is an important part of the protocol in terms of business marketing on Facebook.

4.2 Using Open Graph Protocol

Open Graph protocol asks developer to reiterate information which is likely to exist in the page. For example when a field is marked up with RDFa in Drupal 7, the related semantic markup is directly added to the HTML markup surrounding the field data.

OGP redefines vocabulary terms which have been around for many years:

```
og:image       -> foaf:depiction
og:latitude    -> geo:lat
```

```
og:postal-code  -> vcard:postal-code
og:email        -> foaf:mbox
og:phone_number -> foaf:phone
```

The problem is that existing RDF data which might already be using legacy vocabularies need to add OPG's specific terms if they want to be included in the Open Graph. This is a recurrent problem which happens every time a new big player adopts RDF, it happened with Yahoo! and Google too. RDF datasets end up with duplicate terms for the same semantic and have to add, say `og:postal-code` and `google:postal-code` even though they already have annotated their data with `vcard:postal-code`.

It also has some limitations which would not exist if more standard RDFa markup was used. More specifically, the Open Graph protocol is not able to disambiguate a webpage and all the resources it might describe. In OGP's eyes, the social objects are the pages (HTML documents) and not the real concepts or physical objects people are likely to show an interest in. Let's look at some examples:

- Take a user profile page (typically of type `sioc:UserAccount`) and the real person it describes (`foaf:Person`): what do you mean when you hit the "like" button, is it that you like that Person, or only that particular profile page of that person (say because it has a funny picture). Drupal makes the difference between the two entities in its RDFa markup, but OPG cannot capture that.
- What if you want to like a particular comment on a page, and not the whole page?
- Same goes for a page about a music album and all the songs it contains.

The Web of Data Tim Berners-Lee and the Semantic Web community has been advocating for years is not what the Open Graph protocol enables, we're still at the old document linking stage here.

The Open Graph protocol introduces `og:type`, an alternative to the widely used `rdf:type`. The rationale behind it is to keep the markup consistent in line with their `<@property>` `<@content>` syntax. However, because the `@content` attribute is used, it means it requires a string as the type of object. The first consequences is a limitation in OGP: it is not possible to specify several types for the same object, for example you cannot say that someone is both an actor and director, something which would easily be specified using RDFa's typeof attribute if only we had proper URIs instead of string. Compare the following snippets. Here is what OPG promotes:

```
<meta property="og:type" content="actor" />
```

and this is what a more RDF friendly markup would look like:

```
<meta about="" typeof="og:Actor og:Director" />
```

By using the `@typeof` attribute (a shortcut in RDFa to specify the type of the object you're talking about), you get rid of the single type limitation, and you get to use real RDF classes which look like strings thanks to the CURIE syntax. Another benefit of using RDFa's typeof is that you are not limited to using types defined by OGP, but any type from any namespace such as `foaf: orsioc:.`

You can embed your email adress, phone & fax numbers like so:

```
<meta property="og:email" content="<?php bloginfo('admin_email');
?>"/>

<meta property="og:phone_number" content="+44 123 456 7890"/>

<meta property="og:fax_number" content="+1-415-123-4567"/>
```

Note that we have set `og:emailto` the WordPress admin email. You may want to change this if the admin email is not one for public knowledge.

As we know, local search is on the increase. Open Graph has added location based tags which can be used for things such as Facebook Places:

```
<meta property="og:latitude" content="37.416343"/>

<meta property="og:longitude" content="-122.153013"/>

<meta property="og:street-address" content="1601 S California Ave"/>

<meta property="og:locality" content="Palo Alto"/>

<meta property="og:region" content="CA"/>

<meta property="og:postal-code" content="94304"/>

<meta property="og:country-name" content="USA"/>
```

5. Implementation

5.1 Semantic Markup in *Drupal*

Drupal[2] is an open source content management system (CMS) that makes it easy for developers and end users to create robust data entry forms. The forms are suitable for capturing structured data and flexibly formatting that data in different ways. *Drupal* facilitates the creation of web sites by handling many aspects of site maintenance, such as data workflow, access control, user accounts, and the encoding and storage of data in the database.

Strictly speaking, *Drupal* itself is not a CMS, but a platform in which various modules can be plugged into and combined to shape a **CMS tailored to your needs**. There are modules for storing different kinds of content, to develop content based on various criteria, in order to present content in different ways and for many other purposes.

A site administrator initially sets up the site by installing the core *Drupal* web application and choosing from a large collection of modules that add specific functionality to the site, such as improved user interface, enhanced search, various export formats, extended statistics and so on. Site administrators need a fair bit of technical knowledge to choose and configure modules, but usually do not write code; this is done by module developers instead. After the site has been set up, *Drupal* allows non-technical users to add content and handle routine maintenance of the site.

Each item of content in *Drupal* is called a node. Nodes usually correspond more or less directly to the pages of a site. Nodes can be created, edited and deleted by content authors. Some modules extend the nodes, for example a taxonomy module allows assignment of nodes to categories, and a comment module adds blog-style comment boxes to each node.

With *Drupal* 7, even small sites are capable of exposing their data with RDF. Basic page and article content is exposed by default in any new *Drupal* 7 site. The key to this offering is the RDF Mapping API. With this API, any form field can be mapped to an RDF property and any content type can be mapped to an RDF type.

For instance, if the site has listings of movie artists, the content can have the type `mo:MovieArtist`. The fields can be mapped to `mo:fanpage`, `mo:biography`, and so on.

Any content types that have the mapping defined will automatically expose content using RDFa, which is RDF in HTML attributes. *Drupal* takes care of the formatting of the markup, making it much easier to publish valid RDFa.

One can automatically create/generate RDFa output in the HTML source on a new *Drupal* site. In order to do so, you have to define your own mappings for any content types you have in your *Drupal* database through interface. Just remember that the RDF module must be activated.

[2] http://drupal.org/

5.2 Open Graph Protocol in *Drupal*

An `opengraphprotocol` module for *Drupal* 7 which takes advantage of its new core functionalities such as the use of namespaces in RDFa. OPG requires to add the `og` and `fb` namespaces in the HTML output. This is something which would have required users to hack their theme in *Drupal* 6, but which is only a couple of lines in a *Drupal* 7 modules due to `hook_rdf_namespaces`:

```
<?php
function opengraphprotocol_rdf_namespaces() {
  return array(
    'og'      => 'http://opengraphprotocol.org/schema/',
    'fb'      => 'http://www.facebook.com/2008/fbml',
  );
}
?>
```

The rest of the module adds the Open Graph protocol RDFa markup in the head HTML element of the page: `og:title, og:type, og:url` and`og:image`. Most importantly, taking full advantage of *Drupal's* content types, the module offers a basic mapping interface to define what type of social object you want your content types to be mapped to which is then reflected in the page markup via the `og:type` property. With fields now in core, the module will also output whatever field is recognized as one of the Open Graph protocol properties like `description, image, latitude, longitude, locality, region, email, phone_number, fax_number`. So for instance, if you create a field 'description' (machine `namefield_description`) its content will be marked up with OGP. Similarly you can create a field of type integer 'phone_number' and it will be exported as well. Finally the module adds the Like button for commodity and automatic integration with Facebook.

The Open Graph protocol's not complete, but none of Google or Yahoo! got it right the first time either, and we believe OGP will align with the best practices.

5.3 Semantic Markup and EPiServer

With the rapid adoption of RDFa and the other semantic markup standards (Microformats and Microdata), semantic markup will have a huge impact on the Web CMS. The fact that Google supports all three formats with its “Rich Snippets” technology will be a major driver in the near future.

EPiServer is a content management system (CMS) with closed source and the platform for all FjordNett sites. In *EPiServer*, editors use new classification property types to connect pages to

Concepts being managed in the Web3 Platform. These concepts are exposed on the website and can show links to other *EPiServer* pages and to other related concepts. The concept driven approach creates a more natural way for users to navigate and find content in *EPiServer*. As well as concept pages for humans to navigate by Concepts for EPiServer exposes data endpoints to allow *EPiServer* data to be accessed by machines. All concepts and normal *EPiServer* pages now expose an RDF representation of the data behind them. This allows *EPiServer* content to be consumed easily by other applications using simple web requests. As well as the RDF representations the Web3 data is also accessible via a SPARQL endpoint. This allows the core domain model to be accessed and used by other applications. However, EPiServer has no functionality for semantic markup (built-in Support for RDFa, Microformats, and Microdata).

PART 2

6 Semantic Support to Existing Pages

6.1 Advantage of Existing Semantic Add Tools

To make the existing FjordNett content more useful, there are several opportunities to add structure. Table 6.1 lists the more common information types that one can easily mark up as structured data.

Table 6.1 Most common information types that can be mark up as structured data

Information type	Structured Markup
People and Organizations	`hCard`, RDF `vCard`
Calendars and Events	`hCalendar`, RDF Calendar
Opinions, Ratings and Reviews	VoteLinks, `hReview`
Social Networks	XFN, FOAF
Licenses	`rel`-license
Tags, Keywords, Categories	`rel`-tag

We have seen in Part 1 that adding the structured markup to a web page is fairly simple.

RDFa derives its power from the vocabularies that it is based on, like Friend of a Friend (FOAF), GoodRelations and Dublin Core.

6.2 Sample Vocabularies

There are vocabularies that can be implemented in (X)HTML by using RDFa. Some of these vocabularies are given in the following Table 6.2.

Table 6.2 Some vocabularies in use of the Web

Dublin Core	This metadata element standard for cross-domain information resource description provides a simple and standardised set of conventions for describing things online in ways that make them easier to find.
SIOC	Semantically-Interlinked Online Communities Project is an ontology that expresses the information contained both explicitly and implicitly in Internet discussion methods, such as blogs or forums mailing lists.

FOAF	The Friend of a Friend ontology describes individuals, their activities and their relations to other people and objects. FOAF allows the description of social networks in a distributed fashion.
DOAP	Description Of A Project is an ontology to describe open-source projects.
GoodRelations	GoodRelations is a vocabulary that can be used for many purposes and it is investigated in Semantic web application and traditional search engine. GoodRelations is a multi-syntax data format because it can be published in different formats like HTML, RDFS etc.

In addition, there are many domain specific vocabularies in fields such as tourism, technology, environmental science, chemistry and linguistics. A lot of your data is likely to fit into areas covered by vocabularies in Table 6.1, in which case you can incorporate them in your planning.

Dublin Core (DC)

DC is metadata about different things; such as network references, locations' information, companies' names and contacts numbers etc. DC adds semantic to these things or resources. DC describes the resource by creating a special class of statement, this statement consists of two parts: elements (nouns) such as a title, subject, type, etc. and qualifiers (adjectives). Here is an example of how objects can be described with DC.

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">

 <rdf:Description  about="http://purl.org/DC/documents/notes-cox-
816.htm">
    <dc:title>Tourist Information</dc:title>
    <dc:description> The information centre for tourism, culture
and experiences for the whole Bergen region.</dc:description>

  <dc:date>2011-10-13</dc:date>
  <dc:format>text/html</dc:format>
  <dc:language>en</dc:language>
  <dc:publisher> Bergen Tourist Board</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

Friend of a Friend (FOAF)

FOAF is a vocabulary of persons and their relations. The aim of creating the FOAF ontology was to connect the information that is published by people on the Web, specially the documents that contain “see also”. In other words documents that contain links that refer to other documents, which helps the machine to make use of that information and it gives the computer programs the ability to move through a machine readable web.

Semantically-Interconnected Online Communities (SIOC)

SIOC (Semantically-Interconnected Online Communities) project is an open specification for describing communities using online discussion forums or blogs, leading to what some may term "distributed conversations". At the moment, online communities are islands that are not interlinked, and the SIOC ontology has been proposed to not only link these communities but to leverage data in ways that were previously unknown.

A revised version of our SIOC specification has recently been published at http://rdfs.org/sioc/ns# and it can be used on its own (having a complete set of terms) or in conjunction with other RDF formats such as RSS 1.0 (and 1.1).

GoodRelations

The GoodRelations ontology covers the e-commerce domain, and is often presented as a means of raising web visibility. It has been used by many companies like the large electronics firm BestBuy, prominent search engines like Google and Yahoo!, the e-commerce web site Overstock.com, OpenLink Software and the online technology book store O’Reilly.

We take GoodRelations as an example of vocabularies that can be used in RDFa. GoodRelations ontology covers the reputation needs for E-commerce. GoodRelations can be used for description of business offerings in a precise way. This ontology can also be used for describing the resources and the relationship between them, the data package that has description of products, prices of products, properties of the products, stores, opening and closing hours and mode of payment etc. All these data can be embedded into the web page, which increases in its role the visibility of the web page in search engines.

6.3 Examples

In this section we will illustrate some examples of semantic support to existing FjordNett web pages[3].

6.3.1 Dickens restaurant

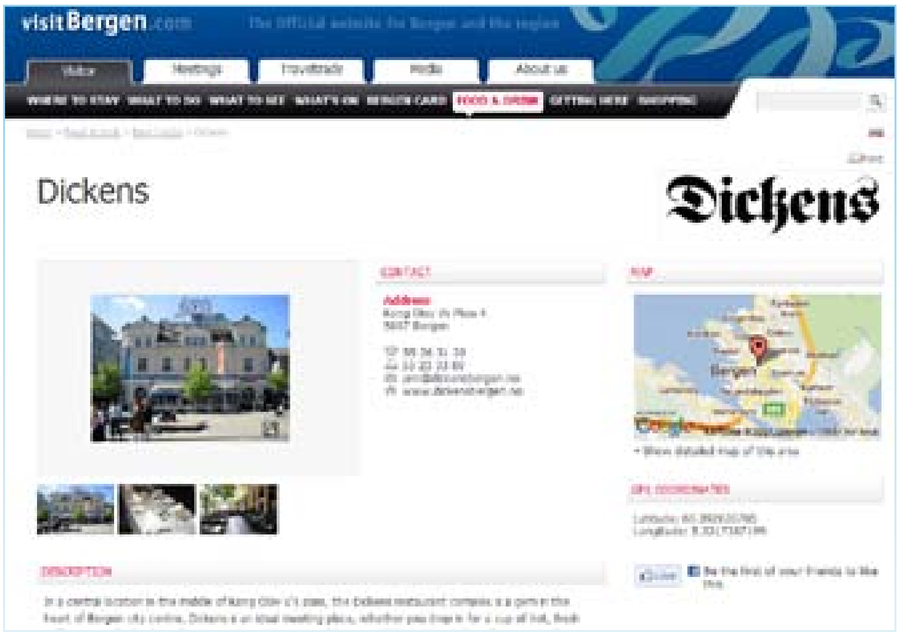


Figure 6.1 Dickens restaurant web page

We will show part of code as an example of using both RDFa attributes and GoodRelations vocabulary on a restaurant web page.

In this example GoodRelations ontology is used to describe the Dickens restaurant web page[4] and name.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
version="XHTML+RDFa 1.0" dir="ltr"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:dc="http://purl.org/dc/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:og="http://ogp.me/ns#"
```

[3] http://www.visitbergen.com
[4] http://www.visitbergen.com/en/RESTAURANTS--NIGHTLIFE/Bars--pubs/?TLp=180498&Dickens

```
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:sioc="http://rdfs.org/sioc/ns#"
  xmlns:sioct="http://rdfs.org/sioc/types#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  xmlns:gr="http://purl.org/goodrelations/v1#"

     xmlns:vcard="http://www.w3.org/2006/vcard/ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

  <div about="#restaurant"
typeof="gr:LocationOfSalesOrServiceProvisioning">
    <div property="rdfs:label" content="Dickens"></div>
    <div rel="vcard:adr">
      <div typeof="vcard:Address">
        <div property="vcard:country-name" content="Norway"></div>
        <div property="vcard:locality" content="Bergen"></div>
        <div property="vcard:postal-code" content="5807"></div>
        <div property="vcard:street-address" content="Kong Olav Vs
Plass 4"></div>
      </div>
    </div>
    <div property="vcard:tel" content="+47 55 36 31 30"></div>
    <div rel="foaf:depiction"
resource="http://media.tellus.no/images/?d=1&p=5443&t=1&.jpg"></div>
    <div rel="vcard:geo">
      <div>
        <div property="vcard:latitude" content="60.392020705"
datatype="xsd:float"></div>
        <div property="vcard:longitude" content="5.3217387199"
datatype="xsd:float"></div>
      </div>
    </div>
<head profile="http://www.w3.org/1999/xhtml/vocab">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<meta about="/node/3" property="sioc:num_replies" content="0"
datatype="xsd:integer" />
```

```
<link rel="shortcut icon" href="http://www.visitbergen.com"
type="http://media.tellus.no/images/?d=1&p=5442&t=1&h=75" />
<meta content="Dickens" about="/node/3" property="dc:title" />

 <title>Dickens</title>

  </head>
<body>

<p>In a central location in the middle of Kong Olav V's plass, the
Dickens restaurant complex is a gem in the heart of Bergen city
centre … … …</p>
… … …</p>
```

6.3.2 Tourist information

Figure 6.2 Bergen tourist information web page

In the following lines, we will semantically annotate the tourist information web page[5] (Figure 6.2). This can be seen as an example of using complex `hCard` + RDFa.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
```

[5] http://www.visitbergen.com/en/Articles/Tourist-Information/

```
          "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">

<html xml:lang="en"
   xmlns="http://www.w3.org/1999/xhtml"
   xmlns:vcard="urn:ietf:rfc:2426#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:w3card="http://www.w3.org/2006/vcard/ns#"
   xmlns:pim="http://www.w3.org/2000/10/swap/pim/contact#"
   xmlns:dc="http://purl.org/dc/terms/">

   <head>
      <title>Tourist Information</title>
      <link rel="foaf:primaryTopic" href="#tourist" />
   </head>

   <body style="max-width:50em">

      <p style="font-style:bold">Book accommodation, activities, fjord
tours, sightseeing, train tickets or the Bergen Card for Bergen and
the region. Currency exchange and souvenirs.
</span></p>

      <div id="tourist_information" class="vcard"
typeof="v:Organisation">

         <img class="photo" alt="Bergen Tourist Information, Torget"
src="http://www.visitbergen.com/ImageVault/Images/id_8209/width_506/c
ompressionQuality_0/conversionFormatType_WebSafe/ImageVaultHandler.as
px" style="float:center;
         margin:1em 0 1em 2em;
         border: 4px solid black;
         />

         <h1 class="fn">Tourist Information</h1>

         <p class="org">
            <span about="#tourist" property="w3card:category"
class="organization-name">Bergen Tourist Information</span>
```

```
          (<span class="organization-unit">The information centre for
tourism, culture and experiences for the whole Bergen region.</span>)
       </p>

       <p class="adr">
         <span class="street-address">Torget 2</span><br />
         <span class="locality">Bergen</span>,
         <span class="region">NO</span>
         <span class="postal-code">5014</span><br />
         <span class="country-name">Norway</span><br />
         <small class="geo" style="color:#999;font-size:67%">From
March 2012 will find Bergen Tourist Information in brand new
facilities in Mathallen, located on Torget.</small>
       </p>
<ul about="#tourist">
         <li class="tel">(+47) 55 55 20 00<span
class="type">work</span></li>
         <li>Visit <a rel="tag foaf:homepage"
href="http://www.visitBergen.com">Our website</a> for more
information.</li>

         <li class="email">info@visitBergen.com</li>
       </ul>

       <h2>We can offer the following:</h2>

<strong><br />Accommodation</strong> <br />At the Tourist
Information, you can book all kinds of accommodation: hotels,
B&amp;Bs, family and youth hostels or private accommodation in a
family home. You can choose between 30 hotels in the city centre and
the rest of the region – ranging from small, charming hotels to big
chain hotels with excellent facilities.</p>

<p><strong>Currency Exchange / Bureau De Change </strong><br />We
have the best opening hours in the city for currency exchange. You
can change most currencies<strong>.</strong> You can also purchase
services and products with foreign currency. </p>
```

```
<p><strong>Tickets for sightseeing, fjord tours and concerts<br
/></strong>We can help you with good suggestions for sightseeing and
harbour excursions in the city, and we sell tickets for all the well-
known round trips and fjord tours that depart from Bergen. In fact,
many tours start right outside our door! </p>

<p><strong>The Bergen Card - practical and inexpensive <br
/></strong>Get free offers and good discounts with the Bergen
Card.&nbsp;&nbsp;&nbsp; </p>

<p><strong>Free brochures <br /></strong>We have all the brochures
you will need for Bergen and Fjord Norway as well as exhibitions that
will guide and inspire you on the rest of your trip. </p>

<p><strong>Train tickets<br /></strong>We sell train tickets for all
rail journeys in Norway.</p>
              <h2>Opening hours:</h2>
       <span property="commerce:hoursOfOperation">
June-Aug: Daily 8.30-22, May and Sept: Daily 9-20, Rest of the year:
Mon-Sat: 9-16</span>
       </div>
  </body>
</html>
```

6.3.3 Concert with Dyvekes Viseklubb

We will present part of the code to illustrate both RDFa attributes and vocabulary on the concert information[6] web page in Figure 6.3. I am using this website to show an example of RDFa usage pattern.

Suppose we have enriched http://www.visitbergen.com/ web site to include event information. Google Rich Snippets are used to mark up information for search engines to use when displaying enhanced search results. We also use some JavaScript code that we found on the Web that automatically extracts the event information from a page and adds an entry into a personal calendar.

[6] http://www.visitbergen.com/en/events/Concerts/?TLp=556680&Concert-with-Dyvekes-Viseklubb

Scenario: Anders finds the http://www.visitbergen.com/ web site through Google and opens the concert’s page. He decides that he wants to go to the concert. Anders is able to add the details to his calendar by clicking on the link that is automatically generated by the JavaScript tool. The JavaScript extracts the RDFa from the web page using RDFa API[7], and places the event into Anders's personal calendaring software — Google Calendar. This is how one can import data.

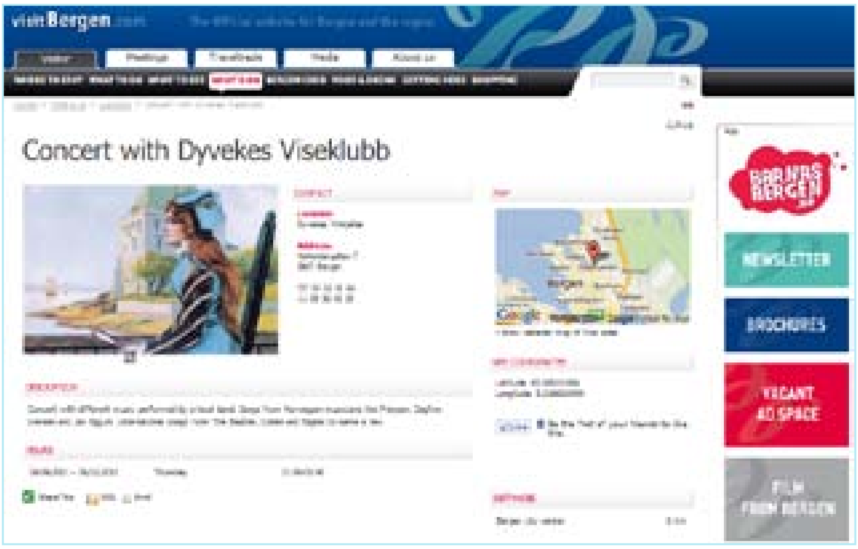


Figure 6.3 concert information web page

```
<div prefix="v: http://rdf.data-vocabulary.org/#" typeof="v:Event">
  <a rel="v:url" href="
http://www.visitbergen.com/en/events/Concerts/?TLp=556680&Concert-with-
Dyvekes-Viseklubb"
     property="v:summary"> Concert with Dyvekes Viseklubb</a>

  <span rel="v:location">
    <a typeof="v:Organization" rel="v:url" href="
http://www.visitbergen.com/en/events/Concerts/?TLp=556680&Concert-with-
Dyvekes-Viseklubb " property="v:name"> Dyvekes Vinkjeller</a>
  </span>
  <div rel="v:photo"><img src="
http://media.tellus.no/images/?d=1&p=9696&t=1&w=350&h=233&.jpg"/></div>
  <span property="v:summary"> Concert with different music performed by
a local band. Songs from Norwegian musicians like Prøysen, Dagfinn
```

[7] http://www.w3.org/TR/rdfa-api/

```
Iversen and Jan Eggum. International songs from The Beatles, Cohen
and Eagles to name a few.</span>
  Hours:
  <span property="v:startDate" content="2009-09-29T21:00">29. Set.
21:00</span>-
  <span property="v:endDate" content="2009-12-29T1:00">29. Dec.
1:00</span>
  Category: <span property="v:eventType">concert</span>

<span rel="v:geo">
        <span typeof="v:Geo">
          <span property="v:latitude" content="60.395101999"></span>
          <span property="v:longitude" content="5.3266533999"></span>
        </span>
      </span>
    </span>
  </span>

</div>
```

How to mark up events for a single venue

If your web page shows information about the venue and then the list of events, then instead of repeating the venue's location in each event markup, you can mark up the venue as an *Organization*, followed by a series of event markups for each event.

The following example shows Organization data for the Dyvekes Vinkjeller, and events scheduled to appear there.

```
<div xmlns:v="http://rdf.data-vocabulary.org/#"
typeof="v:Organization">
  <span class="fn org"> Dyvekes Vinkjeller</span>,
  <span class="adr">
    <span class="street-address"> Hollendergaten 7</span>,
    <span class="locality">Bergen</span>,
    <span class="region">NO</span>
  </span>
  <span class="geo">
    <span class="latitude">
```

```
      <span class="value-title" title="60.395101999" ></span>
    </span>
    <span class="longitude">
      <span class="value-title" title="5.3266533999"></span>
    </span>
  </span>
  <span
class="photo">http://media.tellus.no/images/?d=1&p=9696&t=1&w=350&h=233
&.jpg</span>
</div>

<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Event">
  <a href="http://www.example.com/events/carola_live" rel="v:url"
     property="v:summary">Carola live</a>
  <img itemprop="photo" src="carola.jpg" />
  When:
  <span property="v:startDate" content="2011-11-15T18:00-08:00">
    Nov 15, 7:00PM</span>
</div>

<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Event">
  <a href="http://www.example.com/events/drumming_concert" rel="v:url"
     property="v:summary">Drumming Concert</a>
  <img itemprop="photo" src="drum.jpg" />

  When:
  <span property="v:startDate" content="2011-12-16T19:00-08:00">
    Dec 16, 7:00PM</span>
</div>
```

This example begins with a namespace declaration using `xmlns`. This indicates the namespace where the vocabulary (a list of entities and their components) is specified. You can use the `xmlns:v="http://rdf.data-vocabulary.org/# namespace"` declaration any time you are marking up pages for people, review, event, or place data.

Also on the first line, `typeof="v:Organization"` indicates that the marked-up content describes an organization. Similary, `typeof="v:Event"` indicates that the marked-up content describes an Event. Every property of the Event (such as the summary, event type, and the starting time) is labeled using `property`. The property name is prefixed with v: (`<span property="v:summary">`).

6.3.4 Augustin Hotel

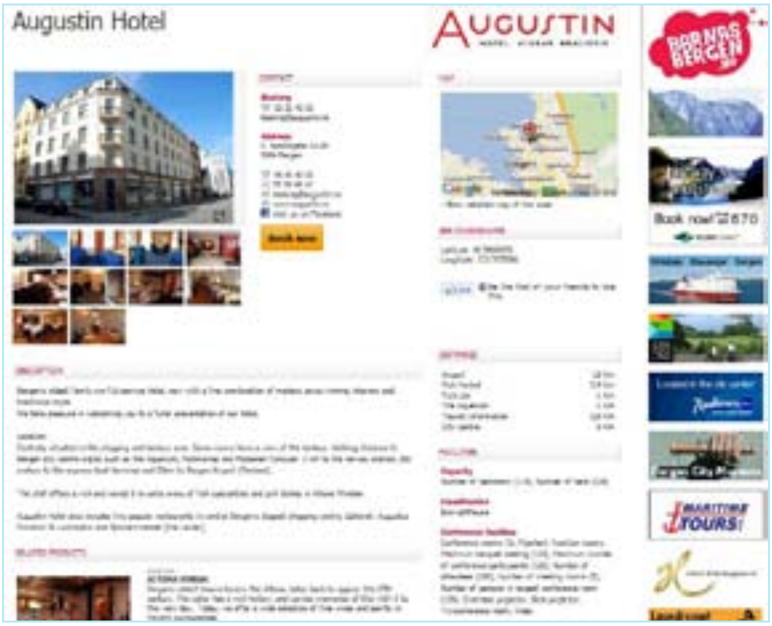


Figure 6.4 Augustin Hotel information page[8]

Any Organization information (for example, details about a hotel, restaurant or office) that is marked up in the body of a web page can help search engines understand location information in reviews or events.

```
<div typeof="vcard:VCard commerce:Business"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:vcard="http://www.w3.org/2006/vcard/ns#"
    xmlns:review="http://purl.org/stuff/rev#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

     <span property="rdfs:label vcard:fn">Augustin Hotel</span>

     <span property="rdfs:comment">
         Bergen's oldest family-run full-service hotel, now with a fine
combination of modern, prize-winning interiors and traditional style.
```

[8] http://www.visitbergen.com/en/Produkt/?TLp=179042&Augustin-Hotel

```
We take pleasure in welcoming you to a fuller presentation of our
hotel... ... ...
     </span>

     <div rel="vcard:adr">
         <div typeof="vcard:Address">
             <!-- If address is not available in parsed form, use
rdfs:label as seen in Example 1 -->
             <span property="vcard:street-address">C. Sundtsgate 22-
24</span>
             <span property="vcard:locality">Bergen</span>
             <span property="vcard:region">NO</span>
             <span property="vcard:postal-code">5004</span>
             <span property="vcard:country-name">Norway</span>
         </div>
     </div>

     <span property="vcard:tel">55 30 40 00</span>

     <div rel="vcard:geo">
         <span property="vcard:latitude"
datatype="xsd:float">60.39602632</span>
         <span property="vcard:longitude"
datatype="xsd:float">5.317675086</span>
     </div>



     <span rel="vcard:photo">
         <img
src="http://media.tellus.no/images/?d=1&p=5015&t=1&.jpg"/>
     </span>

     <a rel="vcard:url"
href="http://www.visitbergen.com/en/Produkt/?TLp=179042&Augustin-
Hotel">Augustin Hotel</a>

     <!-- Review -->
     <div rel="review:hasReview">
```

```
         <div typeof="review:Review">
             Rated <span property="review:rating"
datatype="xsd:float">3.5</span> on a scale of
             <span property="review:minRating"
datatype="xsd:integer">1</span> to
             <span property="review:maxRating"
datatype="xsd:integer">5</span>
             Rated <span property="review:totalRatings"
datatype="xsd:integer">14</span> times.
         </div>
     </div>

     <span property="commerce:hoursOfOperation">Breakfast daily</span>
     <span property="commerce:parkingOptions"> parking lot</span>
     <span property="commerce:attire">Business casual</span>
     <span property="commerce:businessCategory">Hotel</span>
     <span property="commerce:cuisine">Norway</span>
 </div>
```

7. Summary

In this report we have presented the basics of semantic markup and covered the technical details of Microformats, OGP, GRDDL, Microdata and RDFa. GRDDL as a popular markup format which automatically converts microformats and RDFa markup information into RDF triples is also included. Microformats and RDFa can be used to add semantic markups to Web documents, and the semantic markups can be added manually, by site owners or developers.

Yahoo!'s *SearchMonkey*, FaceBook's *OGP* and Google's *Rich Snippets* are applications which take advantage of the added markup information. It is therefore indeed possible to create large-scale applications based on manually or (semi-) automatically created markup information.

Semantic markup gives you the ability to make more information available to the search engines, in a format that is more easily understood, semantically. That's a major plus, from a relevancy standpoint; and thus providing more information to entice the user to visit your site.

Microformats have been established the longest of the three annotations– RDFa, Microformat and Microdata, and used by the search engines the longest. Google and Yahoo! both introduced `hCard` microformat on their own web pages by marking up local listings with it. This was later followed by Yahoo! demonstrating that their crawling and code interpretation services were set up to parse both Microformat and RDFa data. Therefore, the well-established semantic protocol is Microformat, followed by RDFa.

Microformat's main benefit is that it works perfectly in existing HTML code, so using it within a page doesn't require any special tags that might overly restrict one's version of HTML nor cause a page to be invalid code. The shortcoming is that it primarily requires using particular naming conventions of class attributes — retrofitting sites to have `hCard` requires renaming of CSS classes or addition of more tags to add in the specially-named classes. Also, the marked up parts of an address or whatever has to be nested properly for it to work.

RDFa was built a bit more flexible, since it was set up as more purely XHTML — and one key characteristic of XHTML is the "X" part — it's "extensible", meaning we can easily add namespaces for our own purposes without breaking a doc. Thus, RDFa added "`property`", "`role`", and "`about`" attributes for labeling information for machines, and an advantage is that you could introduce them without essentially changing your CSS. The disadvantage is that you technically need to have your document be well-formed to validate under XHTML — a more rigid document coding model than found under earlier versions of HTML. You could add the RDFa portions to your content without making the entire page XHTML-valid, but doing so is risky since it assumes that search engines will be able to properly parse the page when it's not properly formed. While browsers and search engines successfully interpret invalid page code all the time, the risk to twisting up RDFa markup interpretation may be greater if the page is invalid, since the markup is dependent upon the machine properly recognizing the XHTML markup elements and interpreting them correctly.

Microdata is the most recent on the block, and is a format proposed by the W3C as a part of HTML5, and it appears to be heavily influenced by Microformats, adopting a number of the same label names found in `hCard`, `hCalendar` and other Microformats. Microdata is attractive because it's embedded in the new HTML5, but we suggest waiting a bit to go full-fledged into HTML5 in general. So far, there are relatively few web sites using HTML5, so we think there is greater chance for misinterpretation of web pages coded in it on the part of search engines.

In the course of this report, we have also discussed various example of coding of these standards and (flat) vocabulary is also discussed. The flat vocabulary called Facebook's Open Graph Protocol is a variant of RDFa. One can use it in combination with the other semantic mark ups for local search optimization. We suggest using it in combination with one of the protocols Google supports.

We have also discussed implementing semantic markup in two content management systems, namely *Drupal* and *EPiServer* and presented advantages *Drupal* offers in this field. Finally, we have provided some examples of semantic support for existing FjordNett pages.

This works basically for *static* pages which are produced in the CMS. In case of *dynamic* updating, from remote database like tellUs[9], if semantic annotations are not produced in the database or automatically generated locally the annotations are lost.

At this point, we think that RDFa is the most appropriate choice, partly because it is the most versatile and RDFa has ultimately become the preferred medium**. Our suggestion would be to implement RDFa technology now step-by-step, and make it a key part of the next redesign of the FjordNett platform.**

9 http://www.tellus.no/

References

[1]. Akerkar, R. Foundations of the Semantic Web. London: Alpha Science International, 2009.

[2]. D. Connolly, Gleaning Resource Descriptions from Dialects of Languages (GRDDL), http://www.w3.org/TR/grddl/ (last accessed on 08.10.2011)

[3]. Dublin Core Metadata Initiative, http://dublincore.org

[4]. F. Manola, E. Miller, RDF Primer, http://www.w3.org/TR/rdf-primer/ (last accessed on 08.10.2011)

[5]. Microformats, http://microformats.org (last accessed on 08.10.2011)

[6]. Open Graph Protocol http://ogp.me/ (last accessed on 20.10.2011)

[7]. T. Celik and K. Marcs: "Real World Semantics" http://www.tantek.com/presentations/2004etech/realworldsemanticspres.html (last accessed on 08.10.2011)

[8]. Importance of Facebook's Open Graph Protocol for Business Marketing. http://xebidy.com/the-importance-of-facebook-open-graph-protocol-for-business-marketing/ (last accessed on 18.10.2011)

[9]. RDFa in XHTML: Syntax and Processing, W3C Recommendation, http://www.w3.org/TR/rdfa-syntax/ (last accessed on 08.10.2011)

[10]. Tools. RDFa Wiki, http://rdfa.info/wiki/Tools (last accessed on 10.10.2011)

[11]. C. Bizer, R. Cyganiak, and T. Heath "How to Publish Linked Data on the Web", http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

[12]. SearchMonkey: http://developer.yahoo.com/searchmonkey/

[13]. HTML Microdata, http://www.w3.org/TR/microdata/

NCE Tourism – Fjord Norway
Lodin Leppsgt. 2b
NO-5003 BERGEN
NORWAY

Tel: +47 55 30 26 40
Fax: +47 55 30 26 50
ncetourism.com

Følg oss på:
facebook.com/ncetourism

Norwegian Centres of Expertise
NCE Tourism Fjord Norway