Chapter 11
Social Networks and the Semantic Web

In recent years, social networking sites have grown with amazing speed. They are not only the source of vast amounts of user-generated data, but they are also changing our lives in almost every aspect, such as finding old friends, sharing pictures, discussing hot topics and finding jobs, just to name a few.

For us, this presents an interesting question: how are social networking sites related to the Semantic Web? Can the Semantic Web idea help social networking sites in any way? Indeed, with help from the Semantic Web standards and technologies, social networking websites have been operating with new meaning, and as a result, users and content publishers are interacting with each other at much more meaningful levels than have been thought of before.

In this chapter, three major social networking sites are discussed as case studies. We will see how the Semantic Web is changing them, and we will evaluate how close each of them is to the Semantic Web realization. For us, this not only shows the power of the Semantic Web idea, but also can be used as a springboard for ideas for our own development work on the Semantic Web.

This chapter can also be viewed as an extension to Chap. 10. Again, it shows the basic flow of the Semantic Web: to publish structured data on the Web and to make sense out of the published data. We will once again see examples of how semantic markup, even just a little, can take us far.

11.1 Overview of Social Networking Websites

Social networking websites often refer to online communities of people who share interests, activities, backgrounds or real-life connections. Users of social networking websites are often represented by their public profiles, and they interact with

L. Yu, *A Developer's Guide to the Semantic Web*,
DOI 10.1007/978-3-662-43796-4_11

each other through chat, message, e-mail, video, video-chat, file-sharing, blogging and discussion groups.

The following are some examples of popular social networking websites:

- Facebook

Facebook[1] is an online social networking service founded in February 2004. Users participate in Facebook by creating personal profiles, adding other users as friends, exchanging messages and receiving automatic notifications when profiles are updated. In addition, users may join common-interest user groups, organized based on workplace, school or other characteristics.

- Google+

Google+[2] is an online social networking service owned and operated by Google. It was first launched in June 2011, and now it is the second-largest social networking site after Facebook. Users participate in Google+ by making connections to other people, and by having these connections in different circles or categories. For example, friends, family and acquaintances are all different circles. Users can share updates, photos, links, videos, etc., within a circle.

- Twitter

Twitter[3] is an online social networking and microblogging service founded in March 2006. Users participate in Twitter by registering on the site, and then sending and reading short messages limited to 140 characters, which are also called *Tweets*. Tweets are displayed on the user's profile page, and users may subscribe to other users' Tweets, which is often called "following someone on Twitter".

- LinkedIn

LinkedIn[4] is an online social networking service founded in December 2002. It is for people in professional occupations. Users participate in LinkedIn by creating their profiles, and by accepting and sending invitations to other users in order to build connections. Those users who are connected often share similar professional interests or backgrounds. The connections can be used to build up a contact network, possibly for job hunting and business opportunities.

- Pinterest

Pinterest[5] is an online social networking service founded in December 2009. By December 2011, it had become one of the top ten largest social networking

[1] https://www.facebook.com/
[2] https://plus.google.com
[3] https://twitter.com/
[4] https://www.linkedin.com/
[5] https://www.pinterest.com/

services,[6] with 11 million total visits per week. Users participate in Pinterest by creating profiles, saving images and categorizing them on different pinboards. They can also follow other user's boards if they have similar tastes. Popular categories include travel, cars, food, film, humor, home design, art, sports and fashion.

There are other popular social networking websites on the Internet that cannot be covered one by one here. In fact, not all of the social networking website are clones of each other. Each one offers a unique user experience from a different perspective, and does what it does better than any other site.

Users of social networking websites join the sites to keep in touch with their old friends, and to create new online friendships with people who share similar interests and backgrounds. Some social networking websites even offer the opportunity to establish and expand business contacts to find a job.

For the rest of this chapter, we will look at social networking websites from a different perspective: we will study how the Semantic Web technologies can help these sites to work and operate with different meanings. To follow the content of this chapter, it is not necessary that you are active on social networking sites. All you need is your understanding about the idea of the Semantic Web.

11.2 Facebook's Open Graph Protocol

Search engine companies and social networking companies always have a need to understand the content of a Web page. For Google this is important because if the content of a Web page is understood (at least to some extent), more relevant search results can be returned so more traffic will stay with Google. For social networking companies such as Facebook, if the content behind the Web pages you are liking or sharing is better understood, you will be served better, and your needs will be understood better.

One way to achieve this is to add semantic markup to the Web pages. As we saw in Chap.10, if Web content is marked up by using schema.org, Google will be able to extract structured data from the markup and to further generate rich snippets. Rich snippets can help users to understand why a page is included in the result and why it is relevant to the search.

For Facebook, this basic idea remains the same. However, the page content is marked up by using Open Graph Protocol (OGP) terms rather than terms from schema.org, and the extracted structured data is used to understand the specific *kind* of thing that is being liked, and to create a *typed* link back to the content page.

In this section, we start from Open Graph Protocol, and our goal is to understand how Facebook uses the markup, and how this whole idea is related to the Semantic Web.

[6] Sloan, Paul (December 22, 2011). "Pinterest: Crazy growth lands it as top 10 social site". CNET News. Retrieved February 2, 2012.

11.2.1 Open Graph Protocol

Open Graph Protocol (OGP) is a simple and compact schema (a collection of types and properties) that one can use to markup a Web page. It was originally announced by Facebook in April 2010, and currently it supports RDFa as the markup language. It is widely used by IMDB, Microsoft, NHL, Posterous, Rotten Tomatoes, TIME, Yelp and many others, as reported by OGP's official Web site.

Let us focus on the basic types and properties of Open Graph schema in this section. With all these technical details in place, we will be ready to move on to examples and applications.

First, understand that when terms (such as types and properties) from OGP are used to markup a Web page, the markup itself has to be placed by using `<meta>` tags in the `<head>` section of the Web page, and RDFa has to be the markup language. This is actually simpler than the markup for generating rich snippets, where the markup can be embedded everywhere in the page. In addition, the OGP schema itself is also a much smaller set when compared to schema.org (and other possible ontologies).

When marking up the page, the first thing required is to specify the type of object that is being described. This is done by using the `og:type` property. Note `og` is the namespace prefix, which represents the overall OGP namespace given by the following URL,

```
http://ogp.me/ns#
```

In fact, RDFa property `typeof` could have been selected to accomplish the same thing, yet the OGP design team decided to use a property of their own.

Following the Semantic Web standards, the value of *type* property (such as `rdf:type` or RDFa's `typeof`) should be specified by a URI that represents a class defined in some ontology. For example, `schema:Place` defined in schema.org, or `foaf:Person` defined in FOAF ontology.

However, in OGP, `og:type` only takes string as its value, and the available types represented by strings are those that are agreed upon by the community. More specifically, when the community agrees on a specific type, it is added to the so-called *list of global types*. For example, the following statement says the current Web page that is being marked up is about a book:

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

Note the `content` attribute is used to indicate that the value of `og:type` will be a simple text string. Table 11.1 summarizes all the global types OGP currently supports.

Note that any non-marked up Web page should be treated as using `website` as its default `og:type`.

Once a type is specified, properties can be added to describe the type. Regardless of the specific type, three properties are required, as summarized in Table 11.2.

Table 11.1 Global types currently supported by OGP schema

Category	`og:type` value	OGP namespace URI

Music	`music.song`
	`music.album`
	`music.playlist`
	`music.radio_station`
		`http://ogp.me/ns/music#`

Video	`video.movie`
	`video.episode`
	`video.tv_show`
	`video.other`
		`http://ogp.me/ns/video#`

Article	`article`	`http://ogp.me/ns/article#`
Book	`book`	`http://ogp.me/ns/book#`
Profile	`profile`	`http://ogp.me/ns/profile#`
Website	`website`	`http://ogp.me/ns/website#`

Table 11.2 Required properties for all types

Property	Property meaning
`og:title`	The title of the rich object as it should appear within the social graph, can be anything you want
`og:image`	An image URL that should represent the rich object within the graph social graph, can be anything you want
`og:url`	The canonical URL of the rich object, and will be used as its permanent ID in the graph, e.g., `http://www.imdb.com/title/tt0117500/` is one such URL

Table 11.3 Optional properties for all types

Property	Property meaning
`og:audio`	A URL to an audio file to accompany the rich object
`og:description`	A short description of the rich object; will be shown inside the rich object
`og:determiner`	The word that appears before the rich object's title in a sentence. Possible choices are `a`, `an`, `the`, "" and `auto`. If `auto` is chosen, the consumer of the data should choose between `a` or `an`. The default is "" (blank)
`og:locale`	The locale these tags are marked up in. Its format should be `language_TERRITORY`. The default is `en_US`
`og:locale:alternate`	An array of other locales this page is available in
`og:site_name`	If the object is part of a larger website, the name that should be displayed for the overall site
`og:video`	A URL to a video file that complements the rich object

In addition, Table 11.3 shows the properties that can be used for any type, and they are optional.

There are also properties that are applicable for only a specific type. For example, a `book` has an `author`, an `isbn` number, a `release_date` and a `tag`.

These properties are written using an extra : in the markup. For example, `book:author`, `book:isbn`. Some other examples are `profile:first_name`, `profile:last_name`, `article:author`, etc.

We will not cover all the details about OGP schema, and what we have learned so far is good enough for us to focus on our main goal in this section: to understand how OGP is helping Facebook and how it is related to the idea of the Semantic Web. If you need to understand more about OGP schema, the online documentation is always a good place to go.

Understand that there is an OGP ontology that formally defines all the types and properties. List 11.1 shows part of this ontology.

List 11.1 Part of the OGP ontology

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix og: <http://ogp.me/ns#> .
@prefix ogc: <http://ogp.me/ns/class#> .
@prefix bibo: <http://purl.org/ontology/bibo/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

##### PROPERTIES #####

og:url a rdf:Property ;
  rdfs:label "url"@en-US ;
  rdfs:comment "The canonical URL of your object that will be
               used as its permanent ID in the graph, e.g.,
               \"http://www.imdb.com/title/tt0117500/\"."@en-US ;
  rdfs:seeAlso dc:identifier, foaf:homepage ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:url .
og:type a rdf:Property ;
  rdfs:label "type"@en-US ;
  rdfs:comment "The type of your object, e.g., \"movie\".
               Depending on the type you specify, other
               properties may also be required."@en-US ;
  rdfs:seeAlso rdf:type ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:string .
og:title a rdf:Property ;
```

```
  rdfs:label "title"@en-US ;
  rdfs:comment "The title of the object as it should appear
                within the graph, e.g.,  \"The Rock\"."@en-US ;
  rdfs:subPropertyOf rdfs:label ;
  rdfs:isDefinedBy og: ;
  rdfs:range ogc:string .
……

##### DATATYPES #####

ogc:mime_type_str a rdfs:Datatype ;
  rdfs:label "mime type string"@en-US ;
  rdfs:comment "Valid mime type strings (e.g.,
                \"application/mp3\")."@en-US ;
  rdfs:isDefinedBy og: ;
  rdfs:subClassOf xsd:string .
ogc:boolean_str a rdfs:Datatype ;
  rdfs:label "boolean string"@en-US ;
  rdfs:comment "A string representation of a true or false value.
                The lexical space contains: \"true\", \"false\",
                \"1\", and \"0\"."@en-US ;
  rdfs:isDefinedBy ogc: ;
  rdfs:subClassOf xsd:string .
ogc:date_time_str a rdfs:Datatype ;
  rdfs:label "date/time string"@en-US ;
  rdfs:comment "A string representation of a temporal value
                composed of a date (year, month, day) and an
                optional time component (hours, minutes).  The
                lexical space is defined by ISO 8601."@en-US ;
  rdfs:isDefinedBy ogc: ;
  rdfs:subClassOf xsd:string .
……
```

This is actually a very compact and intuitive ontology. The most noticeable feature is that there is no class definition included in the ontology, and that is also the reason why `rdfs:range` is used and `rdfs:domain` is never in use. This agrees with the fact that in OGP, `og:type` only takes strings as its values, as we have discussed earlier. We discuss this again in Sect. 11.2.2.3.

11.2.2 How Does It Work: Creating Typed Links Using OGP

11.2.2.1 The Basic Idea and Process

As we just discussed in Sect. 11.2.1, the Facebook OGP schema is a set of types and properties that one can use to markup a Web page. With the markup in place, the Web page can become part of the Facebook experience, and more importantly, Facebook is now able to tell what specific kind of thing is being liked. Let us take a look at how this is actually implemented.

Let us say a Web page has been marked up using OGP types and properties. In addition, a Facebook "Like" button has also been placed on the same page.

Now a visitor is looking at the page and decides to click the `Like` button. The moment the `Like` button is clicked, Facebook will create a connection between the visitor and the Web page. The web page will show up in the "Likes and Interests" area of that user's Facebook profile, and it will appear as a *rich* object in Facebook's social graph. The richness of the object depends on how much markup has been added. The object that represents the Web page can now be referenced and connected across social network user profiles, blog posts, search results, Facebook's news feed and more.

For Facebook, an even more important result from the added markup is that it is able to tell what exactly is being liked. To see this, consider the case where there is only a `Like` button on the page, and page itself does not contain any OGP markup at all. A visitor can still click the `Like` button; however, all Facebook knows in this case is that a link is liked and nothing further. Unfortunately, for Facebook, all these links are created equal, and there is nothing more exciting that Facebook can offer.

On the other hand, with the OGP markup, Facebook can parse the required `og:type` property from the markup, and comes to the realization that, for example, a book is being liked. It can therefore go out to find others who liked the same book, and may even recommend more books.

What if it is a movie that is being liked? Or, what if it is a song that is being liked? One can easily imagine these similar scenarios. The important fact is, for Facebook, these Web pages are still just links. However, with the added OGP markup, these links have transformed from *untyped* links to *typed* ones, and that is the reason behind all the exciting things Facebook can offer.

11.2.2.2 Open Graph Markup Examples

Before we move on to see what Facebook can do with the added markup, let us first look at one example to understand what we have discussed so far. List 11.2 shows a markup for a book.

List 11.2 Example OGP markup for a `book` object

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:og="http://ogp.me/ns#"
      xmlns:book="http://ogp.me/ns/book#"
      xmlns:fb="https://www.facebook.com/2008/fbml">

<head>
<meta property="og:type" content="book" />
<meta property="og:title" content="A Developer's Guide to the Semantic Web" />
<meta property="og:site_name" content="Amazon.com" />
<meta property="og:description" content="this is a book about the Semantic Web" />
<meta property="og:locale" content="en_US" />
<meta property="og:url"
      content="http://liyangyu.com/book_test0.html" />
<meta property="og:image"
      content="http://www.liyangyu.com/image/devGuide.jpg" />
<meta property="book:author" content="http://liyangyu.com" />
<meta property="book:isbn" content="978-3642159695" />
<meta property="book:release_date" content="2011-01-05" />
<meta property="book:tag" content="the Semantic Web" />
<meta property="book:tag" content="deveoper's guide" />
</head>

<body>
<div id="fb-root"></div>
<script>(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/en_US/all.js#xfbml=1";
  fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));</script>

<!-- other content are not included here -->

</body>

<div class="fb-like"
    data-href="http://liyangyu.com/book_test0.html"
    data-layout="standard" data-action="like"
    data-show-faces="true" data-share="true"></div>
```

Fig. 11.1 A page shows up in Facebook space as a rich object

Note the content on the page (in between `<body>` tag) is not included, since they are irrelevant to the markup, which can only exist in `<head>` section. The markup includes the required properties as shown by the bold lines, together with some optional ones and properties related to `book` type. The `<script>` tag and `<div>` tag are responsible for the `Like` button on the page.

To test this, a public Web server is needed, and you should at least have the right to push static content pages such as the one shown in List 11.2 to the server. The author was able to test this by using his own domain, liyangyu.com. Once this page was published on the server, a browser was used to access the page. The moment the `Like` button was clicked, a link was created between the Facebook profile and the page. The page itself then shows up as a rich object in Facebook page, as shown in Fig. 11.1.

It is obvious how the properties are mapped to the GUI elements in Fig. 11.1. For example, `og:title` is mapped to the title of the book, and `book:author` is mapped to the link text under the title.

Meanwhile, the key information is at the top, which says "Liyang Yu likes a book." `book` is the typed link we are interested in, and it tells Facebook, it is a `book` that we like, not anything else.

For comparison, we can remove all the markups in `<head>` section (keep everything else the same) and test it again. Figure 11.2 shows how it looks in Facebook after the markups are removed.

As shown in Fig. 11.2, although Facebook is able to retrieve some existing markup from the main page and use that information to show the picture and the title of the page, it can only say "Liyang Yu likes a link." This is what is called an untyped link—it does not tell Facebook anything specific, because a link could be a link to a book, to a movie, to an event, to a place, and so on. We will come back to this topic in the next section.

Finally, understand that one can always check the OGP markup by using the debugger[7] provided by Facebook. The URL of the page is needed to run the tool. For example, Fig. 11.3 shows the debug information for List 11.2.

[7] https://developers.facebook.com/tools/debug/

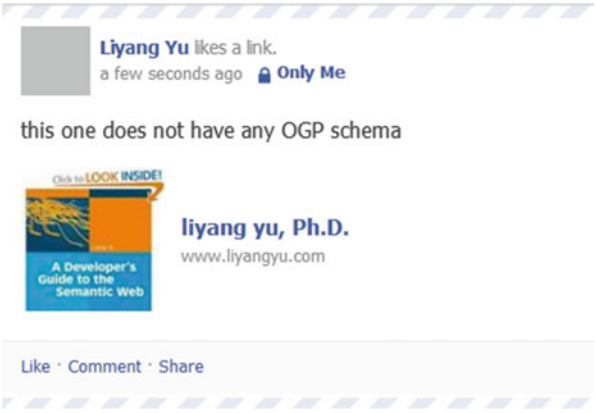

Fig. 11.2 A page shows up in Facebook space as an untyped link

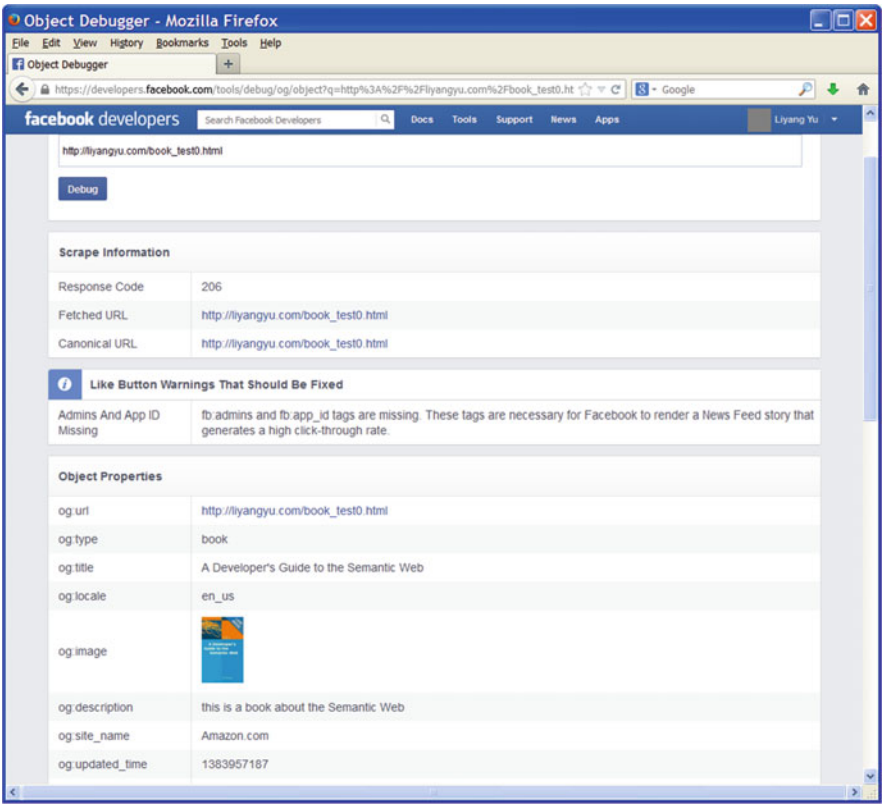

Fig. 11.3 Open Graph Object debugger output for List 11.2

11.2.2.3 Open Graph Issues

At this point, we have learned how semantic markup is added to a Web page for a typed link to be created by Facebook. Based on what we have learned about the Semantic Web, we should be able to identify several issues with this approach.

Facebook supports RDFa as the markup format. However, Facebook's usage of RDFa does not fully agree with the W3C's RDFa guideline. List 11.3 shows what the Facebook design team tried to use.

List 11.3 OGP markup that Facebook design team tried to use

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:og="http://ogp.me/ns#"
      xmlns:dc=" http://purl.org/dc/elements/1.1/"
      xmlns:foaf=" http://xmlns.com/foaf/0.1/"
      xmlns:fb="https://www.facebook.com/2008/fbml">

<head>
   <meta property="og:type" content="book" />
   <meta property="dc:title"
         content="A Developer's Guide to the Semantic Web" />
   <meta property="foaf:logo"
         content="http://www.liyangyu.com/image/devGuide.jpg" />
</head>
```

List 11.3 was believed to be too complex for content developers. For instance, to follow List 11.3, one has to learn FOAF ontology and Dublin Core, at a minimum. The decision then was to follow what we saw in List 11.2, which is also known as the *minimal form* of RDFa.

Perhaps the most surprising fact is that OGP ontology (Sect. 11.2.1) does not define any class, and property `og:type` always has to use a text string to represent the class type. Since there is no class defined, there is no such thing as class hierarchy, and there is no such concept as domain of a property.

This is probably acceptable for the immediate goal set by Facebook. However, moving forward, semantic markup from each page can be gathered together and interesting applications can be built on top of these structured data. When this comes into the plan, the lack of class and class hierarchy can hurt these applications and can greatly limit the possibility of reasoning on the gathered structured data.

For example, movies, books and songs can all be considered as creative work. Assuming there is a need to understand how many users have liked creative work, an application based on OGP schema has to hard-code the fact that liking creative work means to like movies, or books or songs. On the other hand, if the ontology had specified the fact that book, movie and song are all subclasses of creative work, the same application would be able to easily understand how to obtain the requested

information without the need to hard-code anything. In addition, new types as subclasses of creative work can be added to the ontology at any time, and new markups can be added to the page freely; the application itself does not need any change.

Another consequence of not having class definition in OGP ontology is that this has practically made it a fact that the added structured data using OGP schema will only be recognized and understood by Facebook itself, not by any other application. For example, an application outside of Facebook can recognize `schema:book` easily, but it will not be able to understand `content="book"` actually means the object on the page is also an instance of `schema:book`.

Besides the above issues, there are other issues with the approach Facebook has taken. For example, there is only one object allowed on each page, which is again a rather surprising limitation. Another consideration is the true openness of OGP schema itself—for it to be qualified as *open* protocol, it should be developed in an open collaboration with the Web. More specifically, Google, Yahoo! and W3C would have valuable contributions to make.

Overall speaking, at this stage, following W3C standards and getting the related Semantic Web elements correct does not seem to be the highest priority for Facebook. Instead, the priority is to make this as simple as possible so content publishers can participate without too much of a learning curve to conquer.

Although this is only a "limited version of the Semantic Web" implemented by Facebook, it is yet another example of how a little semantics can get us far, as is discussed in the next section.

11.2.3 Implications for the Semantic Web

To put it simply, Facebook, at least within its own domain, has implemented its own version of the Semantic Web. The added markup using OGP schema maps to semantics, that is, the knowledge that the user is not just interacting with a Web page, but is liking a specific kind of thing. Furthermore, the things being liked can be bucketed into categorizes such as books, movies, music, etc. This simple and somewhat flawed (Sect. 11.2.2.3) semantic layer can then give rise to all sorts of applications that are far from simple and ordinary.

The most recent application based directly on the structured data generated by parsing the added markup is Facebook's new *Graph Search*,[8] introduced in January 2013. More specifically, since the added markup, Facebook now knows *who* likes *what*. It is then easy to answer questions such as "my friends who like books," or even "my friends who like books and also live in Atlanta". These are just some example questions that can be easily answered by Graph Search.

[8] https://www.facebook.com/about/graphsearch

Based what we have learned already, the magic behind Graph Search is clear. When a query such as "friends who like book and live in Atlanta" is submitted by a user, Graph Search traverses all of the user's relationships and find his friends who like books, and then filters them by checking whether they are currently living in Atlanta or not. By applying the same graph traversal algorithms, one can see how even more complex questions can easily be answered. To eliminate the confusion of how to perform such complicated searches, Facebook offers a set of drop-lists and controls that are useful to make refinements to the questions one might ask.

The release of Graph Search has triggered quite a bit of discussion around key issues such as privacy, implications for users and publishers, etc. These topics are outside the scope of this book. For us, Graph Search is at least a great example of how added semantic markup can help us to accomplish functionalities that would not be possible otherwise. At this point, we do not know exactly what other applications will be built on top of Facebook's data repository, but we know for sure they will be even more powerful and impressive.

Perhaps the most important takeaway from what Facebook has accomplished should be the following: Facebook has successfully provided a strong incentive for content publishers and Web sites to finally add more markups in their sites. For the world of the Semantic Web, the more markup data that exists, the better the applications one can build.

11.3 Twitter Cards for Structured Information

With what we have learned about Facebook's Open Graph Protocol, it is not difficult to understand Twitter Cards. Again, the basic idea remains the same: add semantic markups into the Web pages. With the added markup, users who tweet links to this content will have a "card" added to the Tweet. This not only means one can control how the content is displayed with Tweets, it also drives more engaged traffic to the site. From this perspective, Twitter Cards is similar to Google's rich snippets.

In this section, we study the details of Twitter Cards. Our goal is to understand how Twitter uses the markup to generate Twitter Cards, and how this whole idea is related to the Semantic Web.

11.3.1 Twitter Cards Overview

In early 2012, Twitter started to experiment with new ways for users to interact with Twitter. The solution Twitter decided upon was to give website owners the option to add markups into their page contents, and when users expand Tweets containing links to these sites, they will see content previews, images, video and more, extending beyond the normal limit of 140 characters.

Table 11.4 Card types supported by Twitter

Card type	Purpose
Summary Card	Summary Card is the default card, which can be used for many kinds of Web content, such as blog posts, news articles, products and restaurants. It includes title, description, and a small image, which are often sufficient enough to provide a helpful description
Summary Card with large image	Similar to the Summary Card, but offers a larger image
Photo Card	Photo card is designed to allow an image to be at the center of a Tweet, implementing the idea of "a picture is worth of a thousand words"
Gallery Card	A Gallery Card can represent collections of photos within a Tweet. This is designed to let the user know that there is more than just a single image at the URL shared, but rather a gallery of related images
App Card	App Card is designed to provide a profile of an application, including name, description and icon. It can also highlight attributes such as the rating and the price
App Installs and Deep-Linking	This is to allow users to download the app (if they don't already have it installed), or deep-link into the app (if the app is ready installed on the user's mobile device)
Player Card	Player Card is designed for steaming media experiences such as audio, video or slideshows. It allows content publishers to present their content inside of an iFrame within the Tweet
Product Card	Product Card is designed to represent retail items on Twitter, and to drive sales. It can showcase a product via an image or a description. It also allows the content publisher to highlight two other key details about the product

Twitter named the technology behind this feature Twitter Cards,[9] and it was first released in mid-2012. At the time of this writing, Twitter supports eight different types of cards, as show in Table 11.4.

When a Tweet links to a content page, if the page has been marked up by using Twitter properties, the Tweet will have a card inside it. The specific type of card depends on the markup in the content page.

Let us use Summary Card as an example to gain more understanding about Twitter property markup. For other card types, the markup process remains the same, with the only difference being the specific card type and related properties for the selected type.

First, understand that all the Twitter properties should be placed inside the `<head>` section of the page, by using `<meta>` tag. Each card type supports and requires a specific set of properties, but all cards share one basic property in common, and that is the card type property, identified by `twitter:card`. And it is always necessary to specify the type of the card. For example, the following markup says the card type is a Summary Card:

```
<meta name="twitter:card" value="summary">
```

[9] https://dev.twitter.com/blog/twitter-cards

Table 11.5 Required properties for Summary Card

Property name	Purpose
`twitter:title`	Title should be concise and will be truncated at 70 characters
`twitter:description`	A description of the page content, as appropriate for presentation within a Tweet. Should not be the same as the title, and should not be used for describing the services provided by the Website
`twitter:image`	An image represents the content of the page, and should be specific to the content, not a generic image such as the logo of the Website

Other possible type values include summary_large_image, photo, gallery, product, app or player. In addition, if no `twitter:card` value is set, Summary Card is the default card type.

For a Summary Card, the required properties are shown in Table 11.5. If any of these required properties are not included in the markup, the card may not be shown in the Tweet.

Other Twitter properties can be used when adding the markup, but they are not required and may be ignored by the Twitter crawler. For other type of Twitter Cards, one can check the online Twitter developer document[10] to find the required properties.

Note Twitter markups are simple key–value pairs, or property–value pairs. The combined collection of property–value pairs defines the overall card experience on Twitter. These simple key–value pairs do not even require a namespace prefix declaration. More specifically, Open Graph markup requires specifying `og` prefix via `<html prefix="og:http://ogp.me/ns#">`; however, no such markup is required for Twitter cards or the "`twitter:`" prefix. Also, note Open Graph markup is implemented by using `property` and `content` attributes, while Twitter cards use `name` and `value` attributes.

In fact, Twitter markup and Open Graph markup are closely related. When checking the page content, Twitter Card processor first looks for markup using Twitter properties, and if none are present, it falls back to the supported Open Graph properties. Besides the `name` and `value` attributes, the Twitter processor can understand `property` and `content` attributes as well.

For example, consider a content page that has been marked up to generate a Summary Card in Tweets. For this purpose, `twitter:description` is a required property. However, if the Twitter processor cannot find this property, it falls back to `og:description` property. Similarly, if it cannot find the required `twitter:title` property, it falls back to `og:title` property. For more examples, Twitter's online markup reference document[11] shows a more complete list of Twitter properties and their corresponding Open Graph Fallbacks.

[10] https://dev.twitter.com/docs/cards

[11] https://dev.twitter.com/docs/cards/markup-reference

11.3.2 How Does It Work: Structured Information for Rich Tweets

11.3.2.1 The Basic Idea and Process

At this point, the basic idea behind Twitter Cards is very clear. A content publisher can add markups by using the Twitter properties, and Twitter will work out the rest. More specifically, any Tweet that includes a link to the page will automatically have a Twitter Card inside it. This gives the reader a preview of the content before clicking through to the website.

Four steps are needed to accomplish this.

- Select the card type.

The currently supported card types were discussed in Sect. 11.3.1, and related details can also be found at Twitter's developer support site. It is the content publisher's responsibility to review the supported card type and decide which one is the best choice for representing the page content.

- Add the pertinent markup to the page content.

The specific markup that is needed in the page depends on the selected card type. Summary Card is used as one example in Sect. 11.3.1 to show the required properties that one has to use. Implementations of other card types can be done similarly. Again, Twitter's online developer document is the best resource to check out the latest properties that Twitter supports.

- Test the markup and submit application.

To make sure the added markup can be recognized by the Twitter processor, Twitter provides a Card Validator[12] for content publishers to check their markup. Content publishers enter the URL of a page that contains markup to start the test. Once the page has been validated, the content publisher can request final approval for the card.

- Tweet the URL and see the card.

After receiving the approval notice from Twitter, the content publisher can Tweet the URL, and should see the card appear below within the Tweet.

11.3.2.2 Twitter Card Markup Examples

Since the potentially long turnover time of application for final approval from Twitter, we will not create Twitter Card markup from the scratch; instead, we will use existing production content page to show markup examples.

[12] https://dev.twitter.com/docs/cards/validation/validator

The existing content page is from the *New York Times*.[13] On the *New York Times*' home page, enter "Roger Federer" in the search box. Click the Search button, and you will find a link in the search result page pointing to an article named *Why Roger Federer is the Greatest of All Time*.

Click the link to open the article.[14] After you enjoy reading this nice piece published on September 10, 2013 by Michael Steinberger, open the page source of this article. On the page source, you will find the Twitter markups. List 11.4 shows part of it.

List 11.4 Twitter markup on one *New York Times* article

```
<meta property="og:type" content="article" />
<meta property="og:url" con-
tent="http://6thfloor.blogs.nytimes.com/2013/09/10/why-roger-
federer-is-the-greatest-of-all-time/" />
<meta property="og:site_name" content="The 6th Floor Blog" />
<meta property="og:description" content="Yes, he has a lousy rec-
ord against Nadal. But he&amp;#8217;s displayed a consistency and
durability at a time when the competition is deeper than it has
ever been." />
<meta property="og:image"
con-
tent="http://graphics8.nytimes.com/images/2013/08/21/magazine/21-
federer-steinberger/21-federer-steinberger-superJumbo.jpg"/>
<meta name="twitter:card" value="summary">
<meta property="twitter:title"
      content="Why Roger Federer Is the Greatest of All Time " />
<meta property="twitter:url"
      content="http://6thfloor.blogs.nytimes.com/2013/09/10/why-
roger-federer-is-the-greatest-of-all-time/" />
<meta property="twitter:description" content="Yes, he has a lousy
record against Nadal. But he&amp;#8217;s displayed a consistency
and durability at a time when the competition is deeper than it
has ever been." />
<meta name="twitter:image"
con-
tent="http://graphics8.nytimes.com/images/2013/08/21/magazine/21-
federer-steinberger/21-federer-steinberger-thumbLarge-v2.jpg"/>
<meta name="twitter:site" value="@nytmag"><meta name="PT" con-
tent="Blogs" />
```

[13] http://www.nytimes.com/

[14] http://6thfloor.blogs.nytimes.com/2013/09/10/why-roger-federer-is-the-greatest-of-all-time/

List 11.4 actually serves as a good example of having both Open Graph Protocol markup and Twitter Card markup on the same page. It is indeed encouraging to see content publishers at the *New York Times* are willing to take the time and effort to learn about markup and have also implemented the markup nicely.

With what we have learned from Sect.11.3.1, List 11.4 does not need too much explanation. Note the Twitter processor is quite tolerant as far as the syntax is concerned. More specifically, one can use the `name` attribute together with the `value` attribute (as recommended by Twitter's developer site), or the `property` attribute together with the `content` attribute. In fact, even the `name` attribute together with the `content` attribute is allowed. The Twitter processor is implemented this way to make sure as many content publishers as possible can participate in marking up their pages without worrying too much about the correct syntax.

Let us now use the Twitter debugger (its URL was given in Sect. 11.3.2.1) to test the article about Roger Federer. Open the validator for Summary Card, click the `Validate & Apply` tab, paste the URL of the page into the text box and click the `go!` button. Figure 11.4 shows part of the validation result.

Note the validator not only collects the markup that it can recognize, but also shows a preview of the card, indicating what it will look like when opened in a Tweet. To include this preview in the validator, a WebKit-based browser has to be used. More specifically, Internet Explorer and Firefox will not work because they

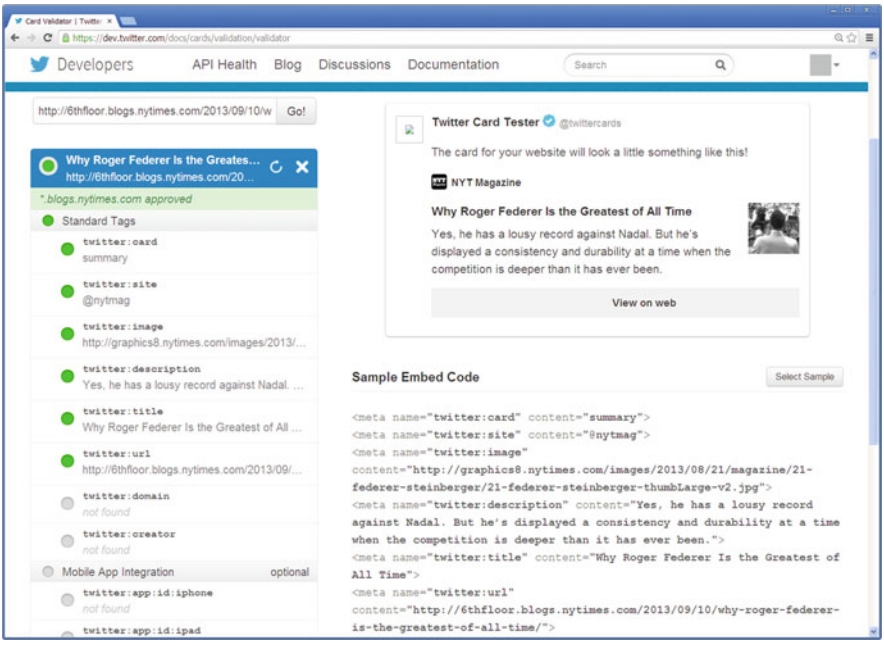

Fig. 11.4 Use Twitter's Summary Card validator to check the Roger Federer page markup

are not WebKit-based, but Apple's Safari[15] browser and Google's Chrome[16] browser will work. Figure 11.4 is based on Chrome browser.

In real practice, after the validation is successfully finished, the next and last step is to apply for the final approval from Twitter. Since we are looking at a production content page that has the markup already and we are also not the owner of the site, it is not possible for us to practice this step. We will therefore simply move on to the next step: Tweet the URL and check the result.

To do so, move to the end of the article. You will find several social media logos, including Facebook, Twitter and Google+. Click the Twitter icon, which will enable us to Tweet the link to this article and share it with our followers.

After creating the Tweet, let us check and see it in my Twitter account. Indeed, there is a new Tweet in my account that reads, "Why Roger Federer Is the Greatest of All Time." Besides the conventional choices such as `Reply`, `Delete` and `Favorite`, the Tweet offers a new choice called `View summary`, with a little card icon attached to it.

This is what Twitter has done on our behalf. More specifically, the Twitter processor has picked up the added markup and it understands these markup are there for a Summary Card. It therefore uses the markup to create a Summary Card attached to the Tweet. Figure 11.5 shows what you will see after clicking `View summary`.

Fig. 11.5 Our Tweet to the article, which has a Summary Card attached to it

[15] www.apple.com/safari/

[16] www.google.com/chrome

Other card types follow the same pattern. For example, if a content page contains Twitter markup for a Photo Card, one will find a `View photo` link under the Tweet that points to this page. Clicking this link opens the Photo Card.

In fact, check your Twitter account and go through the Tweets you have received. It is likely that you will see quite a few different Twitter Card types attached to your Tweets. And now, you understand what is behind all these cards.

11.3.2.3 Twitter Card Issues

We have discussed issues about Open Graph Protocol in Sect. 11.2.2.2. Compared to Facebook's Open Graph, Twitter's Twitter Card idea, as far as its implementation is concerned, is even further away from the Semantic Web standards.

Twitter's markup does not follow any of the popular formats, such as microformats, microdata or RDFa/RDFa Lite. As shown in List 11.4, the markup format is simply a collection of name–value pairs, which at best remotely resembles the minimal form of RDFa. In addition, only one card type can be specified per content page.

Twitter does not have any ontology of its own. The card types, such as Summary Card, Player Card, Photo Card, etc., are created based on the recommendation from the community. Besides a set of common properties that all card types share, there are some specific properties associated with each card type. Similarly, these properties are specified in related developer documentation, and not expressed in the form of a formal ontology. In addition, there is no such concept as URI; thus there is no way to uniquely identify a resource in the real world.

Twitter understands the value of sharing vocabularies. Effort is put into maximizing the interoperability by making the Twitter processor understand at least Open Graph Protocol. In fact, a better choice could have been schema.org from the beginning—the entire Twitter markup needs so far can all be accomplished by using terms from schema.org. As we will see in the next section, Pinterest has decided to use schema.org, which works quite well for their needs.

In fact, because of Google rich snippets (Chap. 10) and some other applications such as LRMI (Chap. 10), Best Buy and Pinterest's rich pins, schema.org has gained great popularity among content publishers. It could have been quite an easy task for publishers to accept schema.org as the markup terms for Twitter Cards as well.

It is probably obvious by now that all the limitations faced by Facebook are also applicable to Twitter Cards. For instance, if new applications were to be developed based on the collected structured data (for marketing and related purposes, for example), these applications would have to be coupled quite tightly with the types and properties defined by Twitter at that particular time point. Future changes to the types and properties would likely force the applications to be changed dramatically, or even to be rewritten. Another example is that the added structured data for Twitter Cards cannot be reused by other applications, unless those applications are

modified and coded so they are able to understand Twitter-specific types and properties.

In summary, as far as the Twitter design team is concerned, the more content publishers who are willing to participate in the Twitter Card markup, the better. Again, simplicity has a higher priority than strictly following the related W3C Semantic Web standards.

11.3.3 Structured Information, But Not Semantic Web Yet

The idea behind Twitter Cards is similar to the idea behind Google's rich snippets: users can get a preview of the content before clicking through to the website. This not only helps the user to understand the content better, but also attracts more engaged traffic to the website.

However, apart from the idea of adding markups to the content, what Twitter has implemented at this point is not the Semantic Web yet, as we discussed in Sect. 11.3.2.3. A good example that can be considered part of the Semantic Web is Best Buy's markup using RDFa.

In fact, it is quite easy to do a comparison. Let us take a Best Buy page, say, http://stores.bestbuy.com/516/, and use `View Page Source` to examine the markup contained in the page. You will find that RDFa standard is strictly followed, and all the terms used are taken from existing ontologies such as schema.org, FOAF, SKOS, vCards, etc. To see the RDF statements that can be extracted from the on page markup, simply paste the page URL into W3C's RDFa 1.1 Distiller and Parser.[17] List 11.5 shows part of these statements.

[17] http://www.w3.org/2012/pyRdfa/

List 11.5 Extracted RDF statements based on the Best Buy page markup

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix r: <http://rdf.data-vocabulary.org/#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix v: <http://www.w3.org/2006/vcard/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://stores.bestbuy.com/516/#pagesections> a skos:Concept;
    skos:narrower <http://stores.bestbuy.com/516/announcements/>,
        <http://stores.bestbuy.com/516/clearance/>,
        <http://stores.bestbuy.com/516/details/> .

<http://stores.bestbuy.com/516/#store_516> a
gr:LocationOfSalesOrServiceProvisioning;
    gr:hasOpeningHoursSpecification
<http://stores.bestbuy.com/516/#storehours_fri>,
        <http://stores.bestbuy.com/516/#storehours_mon>,
        <http://stores.bestbuy.com/516/#storehours_sat>,
        <http://stores.bestbuy.com/516/#storehours_sun>,
        <http://stores.bestbuy.com/516/#storehours_thu>,
        <http://stores.bestbuy.com/516/#storehours_tue>,
        <http://stores.bestbuy.com/516/#storehours_wed>,
        <http://stores.bestbuy.com/516/details/>;
    rdfs:seeAlso <http://stores.bestbuy.com/wp-content/store-
images/516/medium.jpg>;
    geo:lat_long "34.051708, -84.284645";
    v:adr [ v:geo [ v:latitude "34.051708";
                    v:longitude "-84.284645" ] ],
        [ a v:Address,
                v:Work;
            v:email <mailto:Stmgrs000516@bestbuy.com>;
            v:locality "Alpharetta, ";
            v:postal-code "30022";
            v:region "GA";
            v:street-address "975 N Point Dr";
            v:tel "678-339-1321" ],
        <http://deals.bestbuy.com/>,
        <http://stores.bestbuy.com/516/details/>,
```

```
<http://www.bestbuy.com/site/olspage.jsp?id=cat12091&type=page&al
lstores=no&mode=fromResult&storeId=516>;
    foaf:depiction <http://stores.bestbuy.com/wp-content/store-
images/516/medium.jpg> .
<http://stores.bestbuy.com/516/#store_review_516> a r:Review-
aggregate;
    r:itemreviewed [ a r:Organization;
            r:name "Best Buy - Alpharetta";
            r:rating <http://stores.bestbuy.com/516/wp-
content/themes/localstores-
specialselections/images/bazaarvoice/transparent-stars/rating-
4.gif> ] .

<http://stores.bestbuy.com/516/reviews/> r:count "240" .

<http://stores.bestbuy.com/516/#storehours_fri> a
gr:OpeningHoursSpecification;
    gr:closes "22:00"^^xsd:time;
    gr:hasOpeningHoursDayOfWeek gr:Friday;
    gr:opens "10:00"^^xsd:time .

.....
```

As a comparison, using the same tool on the URL that points to the *New York Times* article about Roger Federer will not extract any structured information at all.

The basic idea of the Semantic Web is to publish structured data on the Web and to make sense out of it. To accomplish this, ideally, data should be published by using W3C standards such as RDF/RDFa, OWL, etc., and ideally should follow the Linked Data principles, such as dereferenceable URIs that return RDF data and link to other data.

However, at this stage of the Semantic Web idea, structured data in any form is better than no structured data at all. Twitter at least has shown another great example of how important it is to publish structured data on the Web, and more importantly, it has successfully provided enough motivation for content publishers to do so.

If more and more companies can follow what Twitter has accomplished, content publishers will eventually understand the importance of adding semantic markups to the content, and will eventually make this a necessary step of the whole publishing process. That will eventually offer a solid foundation for the full implementation of the Semantic Web idea.

11.4 Rich Pins for Structured Information

Pinterest's rich pins remind us of Google's rich snippets. Indeed, just like rich snippets, rich pins allow the consumers to know more about the item on the pin and to further make a decision about it without ever leaving the pin itself.

Again, the added semantic markups on the Web pages are what make this possible. In this section, we study the details of rich pins. Our goal is to understand how rich pins are generated by using the markups on the page, and how the idea and implementation of rich pins is related to the Semantic Web.

11.4.1 Rich Pin Overview

Pinterest is currently the most popular social scrapbooking tool on the Web. To make it more interesting and attractive to users, Pinterest has decided to allow publishers to attach structured data to their content, so the pins can be generated with more relevant information, such as where to actually purchase the product, and whether the movie is suitable for children to view, etc.

The technology behind this feature is known as rich pins, and was first released in May 2013.[18] In this section, we focus on the basic pin types and their properties, and exactly what markup is needed to generate rich pins. With all these technical details in place, we will be able to understand examples and its applications.

At the time of this writing, Pinterest supports four different types of rich pins, as shown in Table 11.6.

Let us use product rich pin as an example to see the detailed markup that is needed for Pinterest to generate a product rich pin. For other rich pin types, one can check the online developer guide[19] to finish the markup similarly.

First, note that to create product rich pin, both schema.org and Open Graph Protocol can be used to implement the markup. If Open Graph Protocol is selected as the vocabulary, the entire markup should be placed inside the `<head>` section of the page, by using `<meta>` tag. On the other hand, if the markup uses terms from schema.org, the markup itself cannot be placed inside `<head>` section, but has to be added inside the `<body>` section, together with the real content. In addition, microdata is the markup format that is currently supported when schema.org is used.

Using terms from schema.org to implement markup is slightly more complex than using terms from Open Graph Protocol. However, schema.org markup supports multiple products on the same page, while using Open Graph Protocol means only one product markup per page is allowed. For this reason, let us focus on the markup using schema.org.

[18] http://blog.pinterest.com/post/50883178638/introducing-more-useful-pins

[19] http://developers.pinterest.com/rich_pins/

Table 11.6 Currently supported rich pin types

Rich pin type	Purpose	Supported vocabularies
Product	Product rich pin includes key product information such as product description, price, availability and where to purchase the product	schema.org, Open Graph Protocol
Recipe	Recipe rich pin includes key recipe information such as ingredients, preparation time	schema.org, hRecipe formats
Movie	Movie rich pin includes key movie information such as release date, director, actor(s), content rating (such as PG-13) and review information	schema.org
Article	Article rich pin includes related information about an article, such as title, description of the article and author	Open Graph Protocol

Table 11.7 Required properties for product rich pin

Property name	Purpose
`schema:url`	Canonical URL for the page, where the markup is added
`schema:name`	Product name
`schema:price`	Offer price. Note this property uses class schema:Offer as its domain
`schema:priceCurrency`	Currency code for the price. Note this property uses class schema:Offer as its domain

Similar to Twitter Cards, each pin type supports and requires a specific set of properties. However, type property is required for all pin types. For example, the following markup,

```
<div itemscope itemtype="http://schema.org/Product">
```

tells Pinterest parser the related markups are here for a product type rich pin.

For a product rich pin, the required properties are shown in Table 11.7.

Other properties can be used when adding the markup, but are not required. These include `schema:description`, `schema:brand`, `schema:availability`, just to name a few.

It is actually important to include property `schema:availability`, because it tells the consumers whether the product is still available for purchasing. Note property `schema:availability` uses class `schema:Offer` as its domain, and its value range is class `schema:ItemAvailability`. Class `schema:ItemAvailability` is defined by enumeration, and its instances are summarized in Table 11.8.

Another important tip to remember is to include the site name by using Open Graph term `og:site_name`, since schema.org does not have a site name property. Having the site name showing on the rich pin lets consumers remember your site, and has the potential to attract more traffic to your site as well.

With the understanding of the product rich pin, other types of rich pins can be similarly studied.

Table 11.8 Possible values for product availability

Instance of `schema: Item Availability`	Indication
`http://schema.org/InStock`	Indicates the item for sale is in stock
`http://schema.org/OnlineOnly`	Indicates the item for sale is available only online
`http://schema.org/InStoreOnly`	Indicates the item for sale is available only in stores
`http://schema.org/OutOfStock`	Indicates the item for sale is out of stock
`http://schema.org/PreOrder`	Indicates the item for sale is available for preorder
`http://schema.org/Discontinued`	Indicates the item for sale has been discontinued

11.4.2 How Does It Work: Generating Rich Pins Using schema.org

11.4.2.1 The Basic Idea and Process

The creation of rich pins is similar to that of Twitter cards. A publisher first adds rich pin markups to the page content, where a `Pin it` button is also offered. When a consumer clicks the `Pin it` button on the page, Pinterest's processor reads the added structured data inside the page and uses it to attach some extra information to the pin it creates, thus turning it into a rich pin. For instance, for a product pin, the extra information includes price, availability and store URL. For a recipe pin, the extra information would be ingredient lists and the time it takes to prepare the food.

Four steps are needed to accomplish this whole process.

- Select the pin type.

The currently supported rich pin types were discussed in Sect. 11.4.1. The idea of rich pin is relatively new compared to Twitter Cards; thus only four different types are supported. It is expected that more and more rich pin types will be released by Pinterest soon. It is again the content publisher's responsibility to review the supported pin types and decide which one is the best choice for representing the page content.

- Add the pertinent markup to the page content.

Each rich pin type has its own set of properties. Some of them are required, and the others are nice to have. Product rich pin is used as one example in Sect. 11.4.1 to show the markup requirements. Implementations of other rich pins can be done similarly. Again, Pinterest's online developer document is able provide all the details that are needed.

- Test the markup and submit application.

Similar to other social networking sites, Pinterest provides a Rich Pin Validator[20] for content publishers to check their markup. Once the markup has been validated, content publishers can request final approval from Pinterest for processing and generating the rich pins.

- Consumers click `Pin it` button, and more incoming traffic to be expected.

Once the approval is granted, you will soon notice the richness of the created pin. This is also where change is expected: with the rich pin, consumers can gain more understanding about the item on the pin without even leaving the pin, and more engaged traffic will lead to more successful business transactions.

11.4.2.2 Rich Pin Markup Examples

We will again use an existing production page as one example to show the markup of rich pin. It is the wall clock[21] example that is also used in Pinterest's online developer guide.

On the product page, again use `View Page Source` to open the page source. On the page source, you will find rich pin markups. List 11.6 shows part of it.

List 11.6 Part of the markup in the wall clock product page (edited for formatting purposes)

```
<div id="review_totals" class="pdreview_sidebar fl "
     itemscope=""
     itemtype="http://schema.org/Product">
   <meta itemprop="manufacturer" content="Bai Design"
         itemtype="http://schema.org/Organization">
   <meta itemprop="brand" content="Bai Design"
         itemtype="http://schema.org/Organization">

   <div itemscope=""
        itemtype="http://schema.org/PriceSpecification">
      <meta itemprop="price" content="25.51">
      <meta itemprop="priceCurrency" content="USD" >
   </div>

   <meta itemprop="name" content="Landmark Studio Wall Clock">
```

[20] http://developers.pinterest.com/rich_pins/validator/

[21] http://www.wayfair.com/Bai-Design-Landmark-Studio-Wall-Clock-715.LA-BAI1296.html

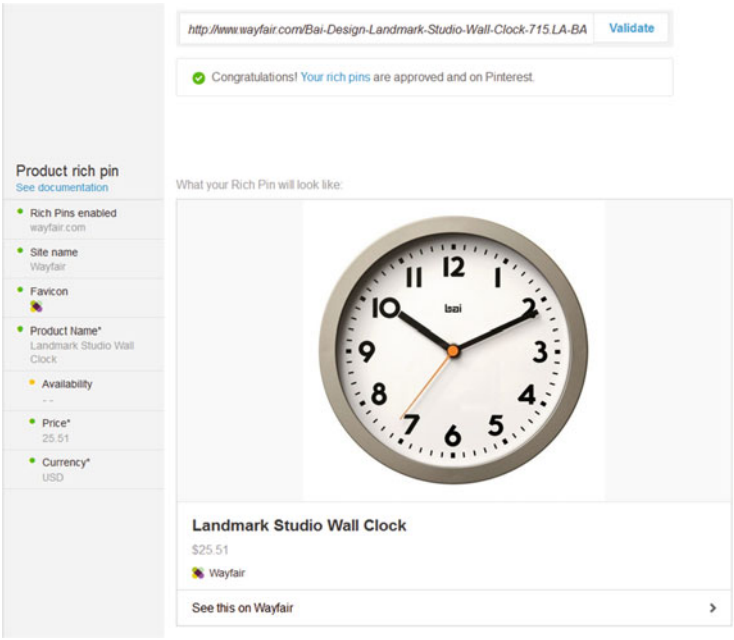

Fig. 11.6 Use Pinterest's rich pin validator to check the wall clock page markup

With what we have learned so far, List 11.6 does not need too much explanation. Note the markup shown in List 11.6 is implemented by using microdata and schema.org. In fact, the same markup is repeated on the same page by using terms from Open Graph Protocol (in the `<head>` section), with `og:url` and `og:site_name` being the extra properties included.

Let us now use the rich pin validator to test the page. Paste the URL of the page into the validator textbox, and click the Validate button to test it. Figure 11.6 shows the validation result.

Note the validator not only shows some of the markup values on the left pane, but also shows a preview of the rich pin. The price, the currency and the store name are the "richness" of this pin. Note the product availability is not included in the markup.

Remember the article in *The New York Times* about Roger Federer (Sect. 11.3.2.2)? In fact, *The New York Times* is one of the partners that Pinterest had when testing the rich pin idea. We can therefore expect to see lots of rich pin markups in their articles. List 11.7 shows the markup for article type rich pin in this Roger Federer article.

List 11.7 Part of the markup in the Roger Federer article in *The New York Times*

```
<meta property="og:type" content="article" />
<meta property="og:url"
content="http://6thfloor.blogs.nytimes.com/2013/09/10/why-roger-
federer-is-the-greatest-of-all-time/" />
<meta property="og:site_name" content="The 6th Floor Blog" />
<meta property="og:description" content="Yes, he has a lousy rec-
ord against Nadal. But he&#8217;s displayed a consistency and du-
rability at a time when the competition is deeper than it has ev-
er been." />
<meta property="og:image" con-
tent="http://graphics8.nytimes.com/images/2013/08/21/magazine/21-
federer-steinberger/21-federer-steinberger-superJumbo.jpg"/>
```

And Fig. 11.7 shows how the article looks in my Pinterest account.

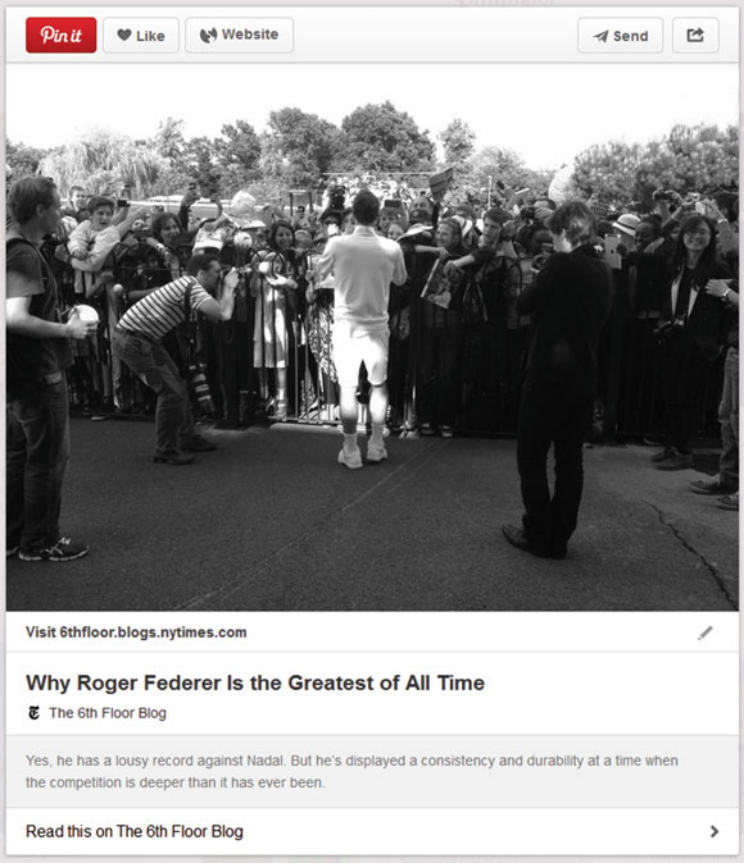

Fig. 11.7 How the article looks after using rich pin markup

Compared to a simple link, this article rich pin tells a lot more about the article itself, without the need to click through it.

11.4.2.3 Rich Pin Issues

Compared to Open Graph Protocol and Twitter Cards, Pinterest's rich pin idea is the closest to the Semantic Web platform—if more restrictions are added.

One of these restrictions is to use schema.org as the *only* markup vocabulary. For now, besides schema.org, Pinterest supports Open Graph Protocol as well. The reason why Pinterest supports Open Graph Protocol is likely because of the consideration of reusing as much existing markups as possible, and also making the markup process as smooth as possible for the content publishers.

However, using Open Graph Protocol terms is a questionable choice, as we have discussed in Sect.11.2.2.3. To position the content well and eventually participate in the Semantic Web global implementation, it is always a good idea to use a true ontology, preferably also a widely accepted one. schema.org, at least at this point, is a far better choice. In addition, because of the fast-growing popularity of schema.org, it is in fact not too much to expect content publishers to become more and more comfortable with schema.org.

Another restriction is to move toward RDFa. Pinterest currently supports microdata, and there is no mention of RDFa or RDFa Lite in their online technical documentation. Again, this is done purposely to avoid any confusion for the content publishers. In Sect. 10.2.2 we discussed the choice between microdata and RDFa, and it is likely that RDFa and RDFa Lite, which are both W3C standards, will offer a smoother transition to the full version of the Semantic Web.

If Pinterest's design team would consider these changes, there would be no other immediate issues that could block Pinterest's great potential.

11.4.3 Semantic Markup at Work

As we have discussed, the idea behind rich pins is similar to the idea behind Google's rich snippets: rich pins let consumers understand the content and the key information before clicking through to the website. This not only helps them to make further decisions, but can also attract more engaged traffic to the website.

More importantly, what Pinterest has implemented so far has positioned itself well for the Semantic Web idea, and the simple implementation of "schema.org + semantic markup" can indeed go quite far. Let us take a look at some examples.

Recall the price information on a product rich pin. What if the price has dropped recently? Similarly, there is also the availability status on the rich pin, whose value can change from "preorder" to "in stock". If you are experienced with rich pins, you probably have noticed that these important information can actually change automatically to reflect the latest updates.

In fact, these automatic updates on the rich pins could be more important then at first glance. Those consumers who have pinned a lamp from a store probably have never shopped at that store, and these pins are often just a wish list for them—they aspire to acquire but are not quite ready to purchase yet. The automatic price drop on the rich pins might just be the push a consumer needs to make the final purchase.

So how does the price and other related information (such as availability status) get automatically updated on rich pins? This is actually the work of the Pinterest crawler: it checks the sites on a daily basis, and it collects the updates to make sure the next time a rich pin is retrieved, the information on that pin will be accordingly updated. The reason why this can be accomplished in a scalable and maintainable fashion is because of the semantic markups contained by the content pages, obviously.

What if the consumer has not recently returned to his pin boards for a while, so he is unaware of the price drop? Pinterest recently released the automatic notification service,[22] where an e-mail will be sent to alert the customer if the price drops. This, again, depends on the fact that Pinterest is able to check and capture the updates in the first place.

For content publishers, the whole process works quite like magic. For those customers who have never shopped at their store and merely pinned a product from the store, the store cannot reach them in any way. However, just because those customers have pinned a product from the store, Pinterest will now be able to reach them and tell them the price change and other related information that can very possibly get them to come to the store.

For a second example, let us consider adding some more information for product rich pin. This is particularly useful for a category such as jewelry, where there is understandably lots of skepticism when it comes to buying online. Without seeing the product in person, consumers often have concerns about the quality and value. If customer reviews from the product page can be added to the markup information, it could be of great help to the potential buyers.

This again is quite feasible for Pinterest, as long as schema.org is used as the underlying vocabulary, which has plenty of terms to describe customer reviews. In fact, one of the rich snippets Google generates has customer review information embedded. It is thus likely to be the case that customer review markups are already readily available on many pages today.

The last example will not be as easy as the previous one and may require more work. More specifically, on a product rich pin, it would nice if *multiple* available prices could be displayed, or, the lowest price and the store that offers this lowest price can be displayed.

Imagine exactly the same product, such as a digital camera, which has a unique model number, is offered by multiple stores, possibly with different prices. Consumers have also pinned the product, each from the web site of a different store.

[22] http://blog.pinterest.com/post/57057851300/pin-a-little-save-a-little

To collect all these available prices and show them on a single rich pin, Pinterest has to be able to identify the fact that these product pins, although created from different store sites, are actually referring to exactly the same item.

What Pinterest currently has in place will not be enough to accomplish this requirement. There are different solutions one could evaluate and use. Whatever the solution is, the idea of using URI to uniquely identify a resource in the world may likely come into play, which is probably why URI is one of the core concepts in the Semantic Web.

Regardless of the detailed solution, it is important to understand that since Pinterest has schema.org and hopefully RDFa as its tool, the following markup would be a good start to address the need:

```
<p vocab="http://schema.org/" resource=URI_of_the_product
Typeof="Product">
```

The same markup, on the other hand, seems to be not quite possible for Facebook and Twitter, at least at this stage.

Without going into any further details, it is not difficult to foresee other examples that can make rich pin experiences even better. It is probably also fair to say, if one follows the Semantic Web standards and tries their best to reuse existing and widely accepted ontologies, there will then be endless possibilities to create enhancements and applications that will bring the user experience to a better and higher level altogether.

11.5 Summary

In this chapter, we focus on how semantic markup (using schema.org and others) can help to change the way social networking sites work. More specifically, we use three most popular social networking sites, namely, Facebook, Twitter and Pinterest as examples, and discuss their related semantic components one by one.

For Facebook, we take a look at the Open Graph protocol; for Twitter, we study Twitter cards, and for Pinterest, we focus on rich pins. These not only show how the Semantic Web can help social networking sites, but also serve as examples to us as developers, so we can draw inspiration and come up with our own applications.