7 Swoogle: A Search Engine for Semantic Web Documents

I am sure you have heard of Swoogle [41] even if you have not used it yet. In recent years, Swoogle has gained more and more popularity in the Semantic Web community, and its idea — to create a system to collect and retrieve Semantic Web documents — has proved to be very useful. In this chapter, we will take a closer look at Swoogle, and you will see the benefit when you finish this chapter.

7.1 WHAT IS SWOOGLE AND WHAT IS IT USED FOR?

Swoogle is often called a Semantic Web search engine. It started as a research project of the Ebiquity Research Group in the Computer Science and Electrical Engineering Department at the University of Maryland, Baltimore County. As the name suggests, its main goal is to provide a search engine for the Semantic Web.

Now, to understand exactly what Swoogle is, we first need to know how Swoogle sees the world of the Semantic Web. For Swoogle, the Semantic Web is a web of Semantic Web documents (SWDs).

What is an SWD? An SWD is defined by Swoogle as an online document written in either RDF or OWL, which then has either `.rdf` or `.owl` as its extension. Swoogle also considers other files as SWDs if they have some other acceptable extensions, such as `rss`, `n3`, or `daml`, to name just a few. The key point is that the Semantic Web, as far as Swoogle is concerned, is a distributed online repository of SWDs.

The developers of Swoogle realized that with the development of the Semantic Web worldwide, there would be more and more SWDs, both ontologies and instance files, physically distributed all over the Web. If we could build a retrieval system that organizes these documents in a systematic way so that both human users and agents/tools can easily conduct searches and queries against this repository, it will greatly facilitate both ongoing Semantic Web and smart agent/tool development.

Swoogle is the product of this vision. It is a crawler-based indexing and retrieval system for Semantic Web documents. More specially (and we will talk about this later), the Swoogle crawler visits the Web to collect SWDs (again, most of these SWDs are files with extensions `.rdf` or `.owl`). For each SWD it discovers, Swoogle extracts metadata from the document and indexes it into an information retrieval system for later searches and queries.

Note that the Swoogle crawler can be viewed as a focused crawler: directed by Swoogle's global view of the Semantic Web (a Web of Semantic Web documents),

this crawler only collects SWDs and ignores all the other documents (HTML, PDF, image files, etc). Therefore, Swoogle can be thought of as a big indexation and retrieval system exclusively dedicated to SWDs, and by the same token, a more precise description for Swoogle should be "Swoogle: a search engine for Semantic Web documents," instead of "Swoogle: a Semantic Web search engine."

First off, to gain a better understanding of Swoogle, let us take a look at how exactly one can use it.

7.1.1 Searching Appropriate Ontologies for Reuse

Remember the reuse idea we have been talking about all the time? When we use an RDF instance document to describe a resource, we always want to reuse the existing URI of that resource if it is appropriate to do so. When doing this, we ensure our newly added description will not be just distributed randomly over the Web, but will be collected and used to facilitate intelligent decisions.

This is true for the ontology: we always want to reuse an existing ontology if it meets our need. If every party decided to invent a unique ontology, there would be no common language and shared understanding about anything, there would be no interoperability of any kind between any two agents, and there would be no global processing possible either. Therefore, without ontology reuse, the very root idea of the Semantic Web is nullified. Reusing ontology, to some extent, is even more important than simply reusing a URI.

This leads to the first major benefit offered by Swoogle: you use Swoogle to find if suitable ontologies matching your need already exist within the underlying domain. You can use specific terms to query the Swoogle engine, and Swoogle will tell you which are the existing ontologies that also use the terms you specified. At this point, you should follow the link provided by Swoogle and check out these ontologies to see if they fit your needs. To this day, this is the most popular usage of Swoogle.

7.1.2 Finding Specific Instance Data

This is probably the other feature you need the most, as it is directly related to resource reuse. You can ask Swoogle to look though the SWDs it collected to find some specific resource, for instance, a friend called `Liyang`, and if there is one, you might want to reuse the URI of `Liyang` and add additional information about this person. In general, you can use Swoogle to query SWDs with constraints on the classes and properties used by them.

Bear in mind that the results returned by Swoogle are normally links pointing to specific documents (with extension `.rdf` or `.owl`, mostly); this is quite different from the traditional search engine, which returns Web pages.

7.1.3 Navigation in the Semantic Web

This is also another important use of Swoogle, and we will see the benefit of navigation in our examples. The key to Swoogle's navigational functionality is the fact that it collects a substantial amount of metadata about SWDs. More specifically,

for each SWD, Swoogle collects the metadata such as the date this document has been discovered, the document type (an ontology or an instance file), whether it is embedded in some other document, whether it is legal (if Swoogle can parse it successfully, it is considered to be legal), etc. Swoogle also collects all the namespaces used by this document, and this is where the navigation capability is provided: you can follow any of these namespaces to find other documents that use this namespace as well; you can then dive into any of those documents, and so on.

This navigational property makes Swoogle an excellent tool to study the structure of the Semantic Web, as the collected metadata — especially the interdocument relations — helps to reveal the internal linkages of the Semantic Web. Based on this information, we can study issues such as how the Semantic Web is connected and how ontologies are referenced.

Swoogle has already been used in several research projects and applications, as reported by the development group of Swoogle [42]. For example, the SPIRE project (supported by the National Science Foundation, NSF) has a team of biologists and ecologists, and the goal of the research is to discover how the Semantic Web can be used to publish, discover, and reuse models, data, and services. To accomplish this goal, the SPIRE project needs to find appropriate ontologies and terms to annotate their data and services. Swoogle's Ontology Search interface, which allows a user to search for existing ontology documents, proves to be of great help to the SPIRE project. For instance, to find an ontology that can be used to describe temporal relations, a user can search for ontologies with the keywords "before," "after," and "interval."

More applications can be found in Reference 42. It is also clear that at the current stage, Swoogle is expected to be used more often by the researchers and developers in the Semantic Web community, as it mainly provides the ability to query and access RDF and OWL documents distributed over the Web. It is also usable by software agents and services through the Swoogle APIs. However, for the casual user who wants to find the best hotel in Las Vegas, Swoogle at its current stage will not be able to help much.

7.2 A CLOSE LOOK INSIDE SWOOGLE

In this section, we will take a look at what is inside Swoogle. Knowing the structure of Swoogle and how it was developed will give us a better understanding of how to use it correctly and efficiently. More importantly, we can view Swoogle as one application category of the Semantic Web itself, and hope the creation of Swoogle will give us ideas for our own killer application on the Semantic Web.

7.2.1 Swoogle Architecture

As mentioned earlier, Swoogle is a crawler-based indexing and retrieval system for the Semantic Web. Its architecture can be broken into four major components:

- SWD discovery component: This component has two distinct Web crawlers that discover SWDs distributed all over the Internet. These two crawlers can be invoked periodically to keep updated information about SWDs. The reason why two distinct crawlers are needed in Swoogle will be discussed in the next section.

In general, there are two different kinds of SWDs considered by Swoogle: Semantic Web ontologies (SWOs) and Semantic Web databases (SWDBs), which is just an instance document. An SWD is qualified to be an SWO when most of its statements are to declare new classes, new properties, and the relationships between classes and properties. On the other hand, if a given SWD mainly introduces new instance data based on existing ontologies, it will be considered an SWDB.

It is true that some SWDs have both class and property definitions and instances. For instance, even a given document that is intended to be an ontology can still have instance data just to make the ontology complete and, similarly, an instance document might have to define several new classes to make the semantics complete. Therefore, there is not always a clear-cut line of demarcation between an SWO and an SWDB. For this reason, Swoogle uses the concept of `ontoRatio`, which is the fraction of individuals recognized as classes and properties.

For example, if Swoogle finds an SWD that defines one class named `Camera` and defines two instances named NikonD70 and NikonD20, then the `ontoRatio` is 0.33. Clearly, this is too low for the document to be viewed as an SWO. Currently, Swoogle will treat an SWD as a strict ontology if its `ontoRatio` is at least 0.8.

- Metadata creation component: The metadata creation component creates metadata for each SWD. The purpose of collecting metadata is to (1) provide information for necessary computation about SWDs and (2) make the search more efficient by providing a navigational tool among all the SWDs collected from the Internet. As we have discussed earlier, this navigational tool can help show many interesting characteristics about the internal structure of the current Semantic Web, which is viewed as a distributed repository of SWDs by Swoogle.
- Data analysis component: This component uses the metadata information to classify the relationship among the given set of SWDs and further calculates the rank of each SWD. Why do we need to calculate the rank and how is it done? We will talk about this shortly.
- Indexation and retrieval component: Swoogle is after all a search engine, and therefore indexation and retrieval are necessary. Details of this component will be discussed later in this section.
- User interface: This is what the user sees when he or she is using the Swoogle search engine. We are not going to talk about it in greater detail; you will see the Swoogle search engine at work in the example section.

In the next few sections, we will take a look at each main component and discuss the technologies used to make them possible.

7.2.2 The Discovery of SWDs

Recall from Chapter 2 that the crawler has to use seed URLs to initiate its journey over the Internet. The same is true for Swoogle. However, instead of providing a set

of seed URLs, Swoogle uses Google to find them. Google provides a set of APIs in the form of Web services, and you can use these services to add constraints to a search and get back the results of the search. Swoogle takes advantage of this API and asks Google to find documents that are potentially SWDs. This is done by asking Google to only find documents ending with `.rdf`, `.owl`, `.daml`, etc., but not those ending with `.jpg` or .html, to name just a few.

Once Google's Web service returns with a set of SWDs, this set will be treated as the seed URLs for Swoogle and the idea is really simple and quite intuitive: if you have found one SWD in some directory, then it is likely you can find more SWDs in the same directory.

Based on this idea, each URL in the seed set is fed to a focused crawler that visits only the Web site that contains the given seed and also the directly linked pages. Because most of the documents on the Web are not SWDs, the idea of a focused crawler works quite efficiently: it often can find more SWDs than Google's estimate.

Besides this focused crawler, Swoogle uses another kind of crawler, called Swooglebot, to find SWDs. The work of the Swooglebot is based on the following observations:

1. URIs are used to uniquely identify the resources and, more importantly, the namespace part of a given URI is likely to be the URL of an SWD.
2. `owl:imports` normally links to another SWD.
3. `rdfs:seeAlso` often links to another SWD.

Therefore, for a given SWD that is discovered by the focused crawler, the Swooglebot uses Jena APIs to parse it, both to confirm that it is indeed an SWD and also to find the URIs and the links associated with `owl:imports` and `rdfs:seeAlso`. Swooglebot then follows these discovered links to search for more SWDs.

The last method Swoogle uses to discover SWDs is to allow users to submit SWDs. It provides a Web-based form to collect these submissions and as you might have guessed, these submitted documents can also be used as good starting points for the Swooglebot.

So, how many SWDs are discovered by Swoogle? As reported by Swoogle's developers [2], as of July 2004, it had found over 5×10^5 SWDs. At the time of my revision (February 2007), the Swoogle Web site reported that it had collected 11.7×10^5 SWDs.

7.2.3 The Collection of Metadata

Swoogle's metadata collection component collects three different types of metadata: the document metadata, the content metadata, and the relation metadata. Each of these different types serves a different goal.

Document metadata includes information such as the URL of the given SWD, the extension, the last modified date, the date the SWD is discovered, etc. Many of these properties are not directly related to search, but can be useful for other components. For instance, the crawler does not need to analyze an SWD if its last modified date has not changed since the last time it was collected.

The content metadata is collected mainly for the calculation of the `ontoRatio`, and as we have discussed earlier, `ontoRatio` is a number to indicate whether a given SWD can be considered as an ontology or an instance document.

Swoogle has identified three interdocument relations. More specifically, one SWD can import another SWD, use a term defined by another SWD, or define a new term using terms defined by another SWD. The metadata collection component continuously collects this relation information among SWDs and saves this information in its related database. This information will serve two purposes: (1) to provide a navigational tool to the user to find the desired document in a more efficient manner and (2) to provide enough information to rank the page and term.

7.2.4 The Calculation of Rankings Using Metadata

The metadata information collected by the metadata component provides the information necessary for Swoogle to rank each document. But first, why is ranking needed?

When a user types a term to find all the ontologies that have defined it as a class or property, it may well be true that many ontology documents have defined this term. Therefore, when returning these documents to the user, which document should be listed first? Similarly, when the user searches for an instance data, several documents may contain the instance; then, which document should be displayed first?

Therefore, a method is needed to measure the relative importance of each Web document, and when the search engine returns results, the document with the highest importance measurement should be returned first. This is the basic idea behind the ordering of the pages returned as results.

PageRank was introduced by Google in order to calculate this measurement. The developers of Swoogle, inspired by this PageRank method, developed their own ranking algorithms called OntoRank and TermRank. OntoRank is designed to evaluate the relative importance of an ontology document, and TermRank is used to sort the RDF terms returned by a term search query. Let us not get into the details of these algorithms in this book; if you are interested, you can check out the related publications on Swoogle. The point is that Swoogle does include a component that can evaluate the importance of each page and term, and this evaluation is done based on the metadata collected by the crawler.

7.2.5 The Indexation and Retrieval of SWDs

Clearly, for the purpose of Swoogle, i.e., to search in the repository of SWDs, indexation is absolutely necessary. The key problem is what should be indexed: a full text index or some selected keyword index?

A traditional search engine, as we have seen in Chapter 2, normally conducts a full text index. A given document is viewed by a traditional search engine as a collection of words (a stream of words). In the case of SWDs, things are different because of the special structure of SWD: any given SWD can be reduced to a set of triples and each triple is made up of three URIs. Therefore, the URI plays the role of the word in a traditional text document, and a given SWD can be viewed as a collection of URIs.

A direct solution is then to take an SWD, represent it using triples, find all the URIs, and use these URIs to index the SWD. However, implementing the indexation using URIs requires the user to input the search term using URIs. For example, if I want to know if any ontology document has defined the class `Camera`, I would have to type the following as the searching keyword:

```
http://www.some_namespace#Camera
```

This is clearly not possible, as the namespace is normally unknown. In fact, I would hope that just typing the word camera will lead me to the correct ontology documents, if there are indeed some that have defined the class `Camera`.

To correct this problem, the developers of Swoogle exploited the fact that any URI, regardless of whether it represents a class, property, or individual instance of a class or property, always has the following form:

```
namespace + localName
```

Therefore, a better idea is to partition each URI into a `namespace` part and a `localName` part, and index on both parts. Now, I can simple type the word "Camera" and find what I want without having to figure out a namespace for the search to work with.

In fact, to make the indexation even more comprehensive, Swoogle also parses the string values of `rdfs:label` and `rdfs:comments` tags; the words in these strings are also used as keywords to build the index system. It is not hard to imagine the cases where this gives more flexibility to users.

By now, we have finished discussing the main components of Swoogle. This should give you a clear picture about why Swoogle works as it does. Let us now take a look at some real-world examples in the next section in order to see exactly how Swoogle can help us with Semantic Web development.

7.3 EXAMPLES OF USING SWOOGLE

Let us take a look at how to use Swoogle to search what we want. Keep in mind that Swoogle is still under development and its crawlers are constantly working, so by the time you try these examples we are going to discuss, you might see different results.

Here is what we want to accomplish by using Swoogle:

1. Use a common term, such as person, to see if there are any extant ontology documents that have already defined a class called `Person`.
2. Take one such ontology (suppose there are ontology documents that have a class called `Person` defined) and find an instance document that uses this ontology.

We can do the following:

Step 1: Go to `http://swoogle.umbc.edu/` to start the search by typing "person" in the search box, as shown in Figure 7.1.

FIGURE 7.1 Search ontology documents that have defined a class called `person`.

After typing in the keyword "person," let us start the search. We will get the results back as shown in Figure 7.2.

Take a look at the first ontology, `http://xmlns.com/foaf/0.1/index.rdf`, by clicking it, and we can see that it does define a class called `Person`, as shown in List 7.1.

LIST 7.1
A Document that Defines the Class "Person"

```
<rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Person"
            rdfs:label="Person"
            rdfs:comment="A person."
            vs:term_status="stable">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="http://xmlns.com/wordnet/1.6/Person"/>
  </rdfs:subClassOf>
  ... ...
</rdfs:Class>
```

Also, remember Swoogle's indexation component we discussed in the last section? The URI of class `Person` is `http://xmlns.com/foaf/0.1/Person`. Therefore, the local name, `Person`, is used to index this ontology document.

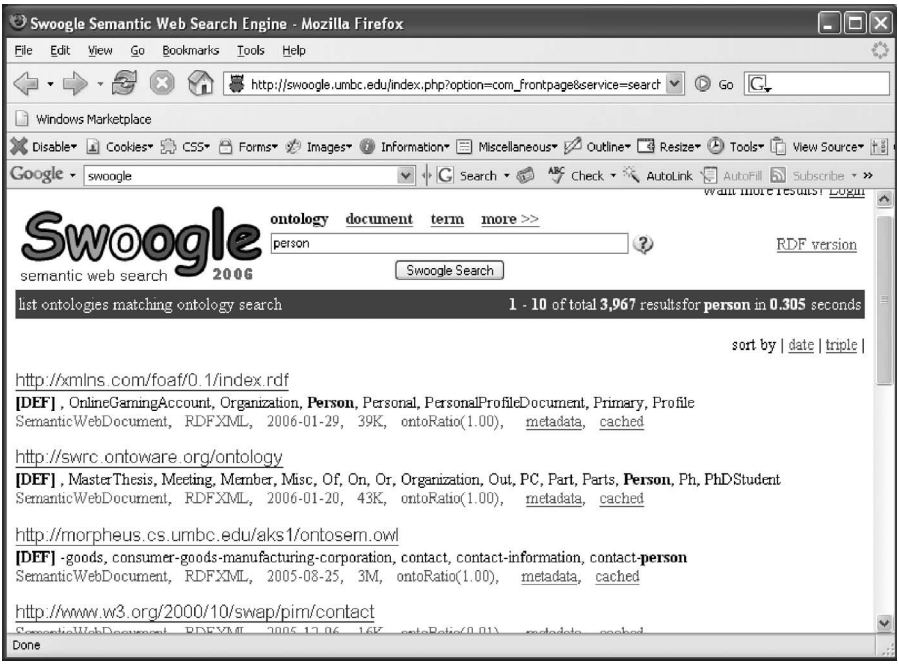

FIGURE 7.2 Search result for keyword "person."

Also, note that `rdfs:label = "Person"`; therefore, using `rdfs:label` to index this document again causes it to be indexed under the keyword "person." This is why we can locate this document by simply typing the keyword "person."

Now that we have confirmed `http://xmlns.com/foaf/0.1/index.rdf` is the ontology we want, let us find an instance document that actually uses this ontology. To do this, let us continue with step 2.

Step 2: Ensure that you are on the result page as shown in Figure 7.2; now click the metadata link associated with `http://xmlns.com/foaf/0.1/index.rdf`. The result is shown in Figure 7.3.

Recall that metadata is collected by Swoogle to provide a navigational tool for the user. Now we have just started our navigation, and you will see it is quite useful for our goal. Figure 7.3 shows the metadata about `http://xmlns.com/foaf/0.1/index.rdf`. For us, we can continue our navigation by following the `related namespaces` link in Figure 7.3.

Step 3: Click the related namespaces link; the result is shown in Figure 7.4.

Now we have navigated to all the namespaces used by the selected ontology. By reading the selected ontology, we know that the `Person` class is defined in the first namespace, i.e., `http://xmlns.com/foaf/0.1/`, which is also the namespace that `http://xmlns.com/foaf/0.1/index.rdf` is defined in. Therefore, we can continue our navigation by following this namespace.

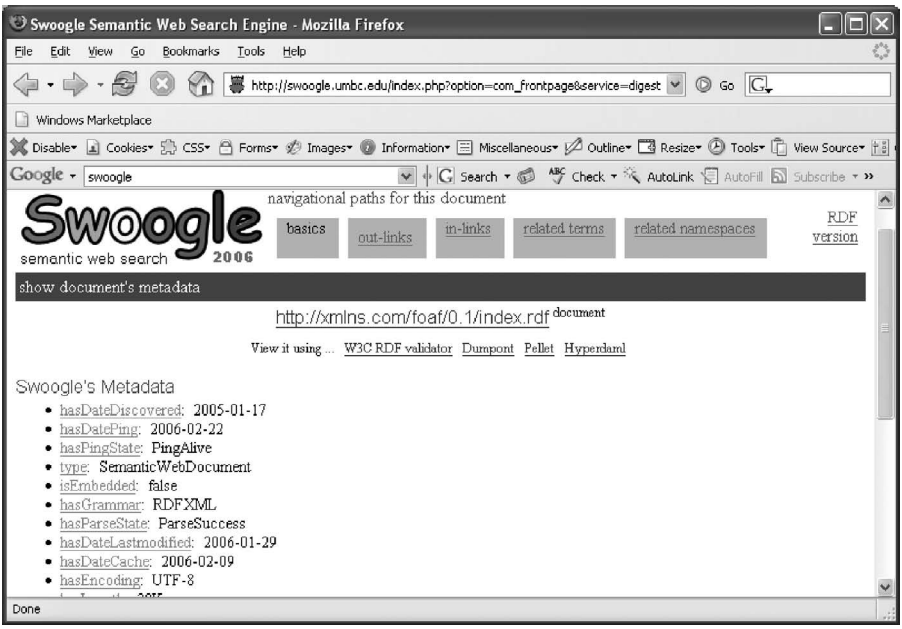

FIGURE 7.3 Metadata of the selected ontology.

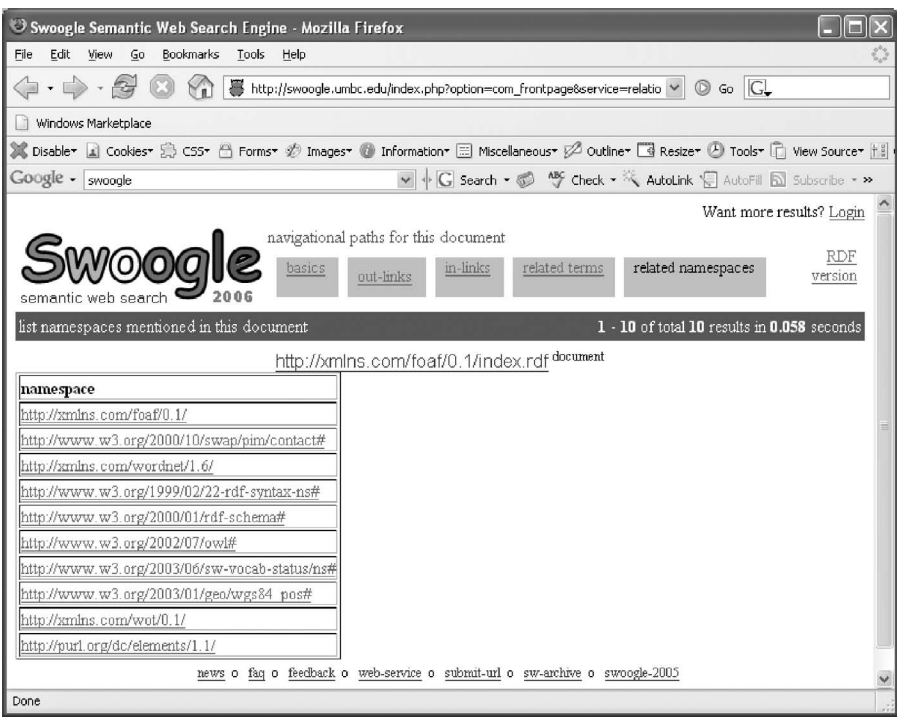

FIGURE 7.4 Navigate to all the namespaces used by the selected ontology.

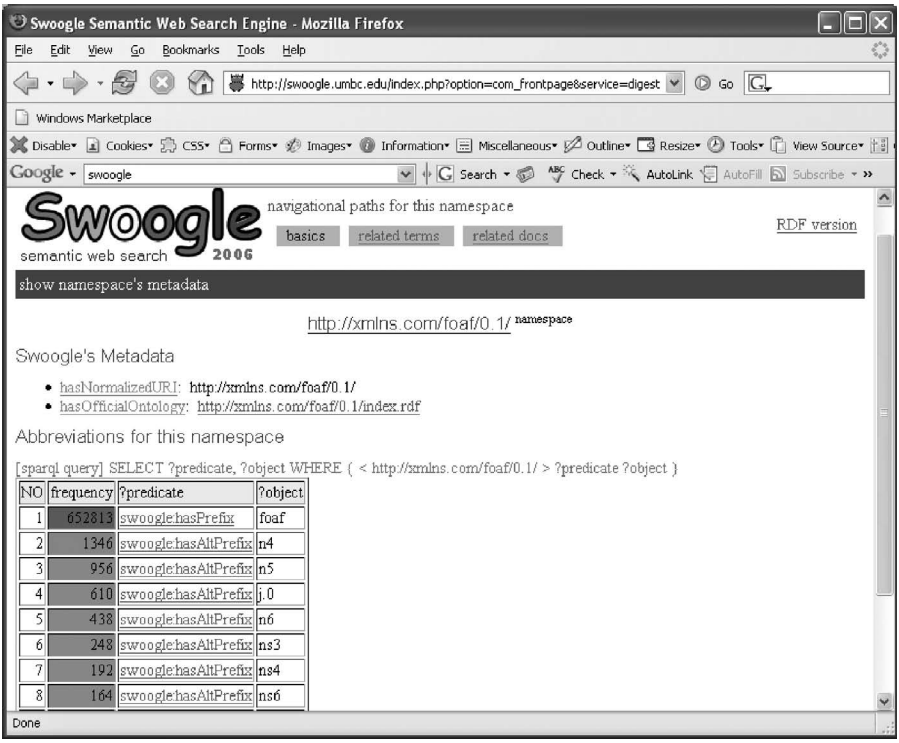

FIGURE 7.5 Navigate to the description of the namespace.

Step 4: Click the first namespace and result is shown in Figure 7.5.

Keep in mind that our goal is to find an instance document that actually uses this ontology. Here is the clue: if any document wants to use this ontology, it has to include this namespace. This leads us to step 5.

Step 5: Click the related docs link on the page shown in Figure 7.5.

The result is shown in Figure 7.6, which lists all the documents that use the given namespace. Remember, the fact that a given instance document is using this namespace does not mean this document has to use the class `Person`; there are many other classes and properties defined in this namespace and the document could just be using them. Therefore, we do need to click several of these documents to find the one that does use class `Person`.

We finally find one such instance document that is using class `Person`. It has the URL `http://www.livejournal.com/community/lj_dev/data/foaf`, and List 7.2 shows the part of this document where the class `Person` is used.

Mission accomplished: we have found an ontology that has defined a class `Person`, and we have also found an instance document that makes use of this class.

This is just an example to show how you can use Swoogle. There are many other ways to conduct search in Swoogle. I will leave that for you to explore, but this example shows one of the most popular uses of the system.

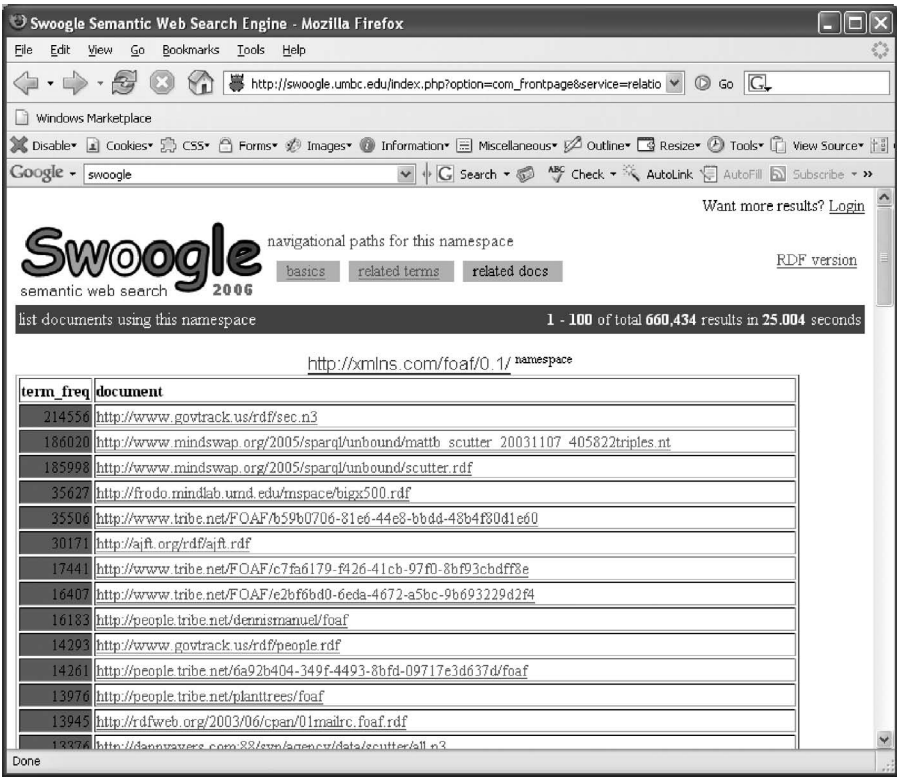

FIGURE 7.6 Instance documents that have used the given namespace.

LIST 7.2
A Document that Creates an Instance of Class `Person`

```
<?xml version='1.0'?>
<rdf:RDF
   xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/">

   ...

   <foaf:Person>
      <foaf:nick>ahm</foaf:nick>
      <rdfs:seeAlso
            rdf:resource="http://ahm.livejournal.com/data/foaf"/>
      <foaf:weblog rdf:resource="http://ahm.livejournal.com/"/>
   </foaf:Person>
```

It is time to move on. In this chapter, we studied a search engine for Semantic Web documents. It is again a chance to practice what we have learned and see a real-world application involving the Semantic Web. In the next chapter, we will study another popular real-world example of the Semantic Web called FOAF, and you will see another flavor of the Semantic Web at work.