CHAPTER 1

Preparing to Program a Semantic Web of Data

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

—Tim Berners-Lee

Welcome to Semantic Web programming—a powerful way to access, use, and share information. Our approach gets you programming quickly through hands-on, practical examples. We maintain a programmer's perspective, not a philosopher's perspective, throughout the book. We focus on applying the Semantic Web to real-world solutions rather than long justifications and explanations.

First, we need to establish a Semantic Web programming foundation. This foundation orients you to this new technology with its jargon and its attitude. The foundation also provides a justification for your learning investment, an investment we do not take lightly.

Our approach and examples come from years of building Semantic Web applications. Our applications employ the Semantic Web to make useable sense out of large, distributed information found throughout the World Wide Web.

The objectives of this chapter are to:

- Form a useful, pragmatic definition of the Semantic Web
- Identify the major components of a Semantic Web application and describe how they relate to one another
- Outline how the Semantic Web impacts programming applications
- Detail the roadblocks, myths, and hype regarding the often misunderstood and misused term *Semantic Web*

- Understand the origin and foundation of the Semantic Web
- Gain exposure to different, real-world solutions that employ the Semantic Web

Semantic Web programming introduces many new terms and approaches that are used throughout the book. This chapter offers a preliminary definition, one on which each chapter expands.

The concept map in Figure 1-1 outlines the chapter. Two main legs establish the key areas: the Semantic Web and Programming in the Semantic Web.

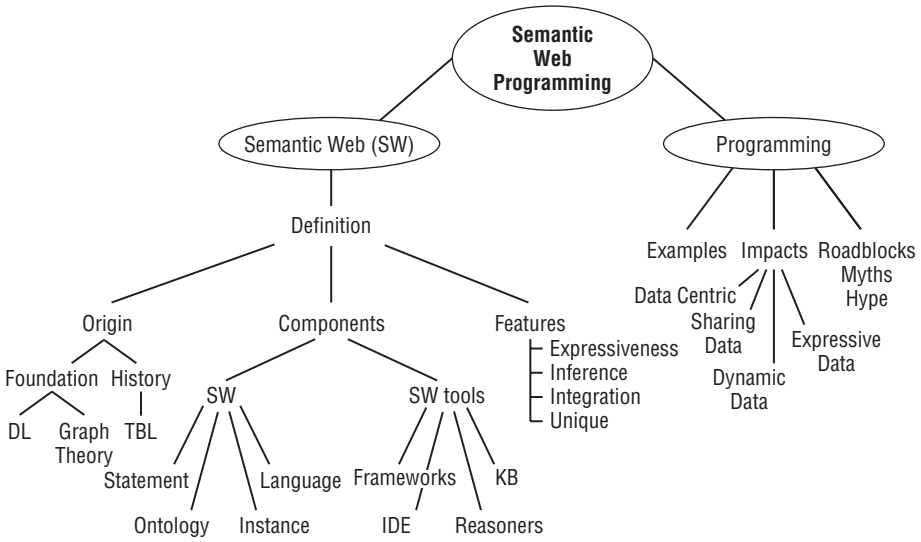

Figure 1-1 Semantic Web concept map

We start with the definition leading to the Semantic Web's components, features, and origins. Then we examine its programming implications.

Defining the Semantic Web

A definition for the Semantic Web begins with defining *semantic*. Semantic simply means *meaning*. Meaning enables a more effective use of the underlying data. Meaning is often absent from most information sources, requiring users or complex programming instructions to supply it. For example, web pages are filled with information and associated tags. Most of the tags represent formatting instructions, such as `<H1>` to indicate a major heading. Semantically, we know that words surrounded by `<H1>` tags are more important to the reader than other text because of the meaning of H1. Some web pages

add basic semantics for search engines using the <META> tag; however, they are merely isolated keywords and lack linkages to provide a more meaningful context. These semantics are weak and limit searches to exact matches. Similarly, databases contain data and limited semantic *hints*, if well-named tables and columns surround the data.

Semantics give a keyword symbol useful meaning through the establishment of relationships. For example, a standalone keyword such as *building* exists on a web page devoted to ontologies. The <META> tag surrounds the *building* keyword to indicate its importance. However, does *building* mean constructing an ontology or ontologies that focus on constructing buildings? The awkwardness of the previous sentence points out the difficulty in simply expressing semantics in English. Semantics are left for the human reader to interpret. However, if the keyword relates to other keywords in defined relationships, a web of data or context forms that reveals semantics. So *building* relates to various other keywords such as *architect*, *building plans*, *construction site*, and so on—the relationships expose semantics. If a formal standard captures the arrangement of terms, the terms adhere to specified grammar rules. It is even better if the terms themselves form an adopted standard or language. The two standards together, grammar and language, help incorporate meaning, or semantics. As this contextual web of grammar rules and language terms expands through relationships, the semantics are further enriched.

The Semantic Web is simply a web of data described and linked in ways to establish context or semantics that adhere to defined grammar and language constructs.

Programmatically, your application can add semantics through programming instructions; however, there is no formal standard for such programmed semantics. In addition, aggregation, sharing, and validation are usually difficult or not possible. The semantics are lost in a maze of if/else programming statements, database lookups, and many other programming techniques. This makes it difficult to take advantage of this rich information or even to recognize it all. The nonstandard, dispersed way of programmatic semantic capture places restrictions on it and makes it unnecessarily complex, essentially obfuscated. Standing alone, the meaning of various terms such as *building* is simply lost.

The Semantic Web addresses semantics through standardized connections to related information. This includes labeling data unique and addressable. Thus, your program can easily tell if this *building* is the same as another *building* reference. Each unique data element then connects to a larger context, or *web*. The web offers potential pathways to its definition, relationships to a conceptual hierarchy, relationships to associated information, and relationships to specific instances of *building*. The flexibility of a web form enables connections to all the necessary information, including logic rules. The pathways and terms

form a domain vocabulary or ontology. Semantic Web applications typically use many ontologies, each chosen for a required information area. The applications can choose to standardize on specific ontologies and translate to ones employed by other applications. Advanced Semantic Web applications could automatically align vocabularies using advanced information techniques that logically employ the many paths within the Semantic Web. Thus, the rich relationships and the many relationship types each contribute to establish semantics—the Semantic Web.

Figure 1-2 illustrates the difference between a stranded keyword, plane, and a Semantic Web of data related to the keyword, plane. The figure uses a graph perspective for easier visualization of the relationships.

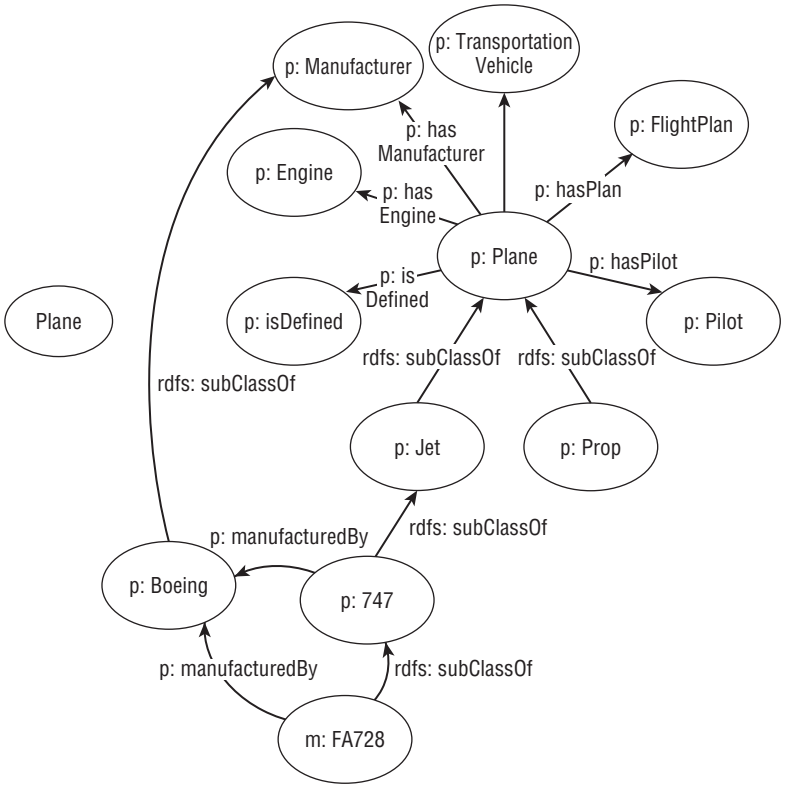

Figure 1-2 Isolation versus the Semantic Web

Shortly we will outline all the major components of the Semantic Web. For now, the fundamental building block of the Semantic Web is a *statement*. This might sound too generic and basic, but this simplicity creates many possibilities. Throughout the book, we explore all types of statements contained in the Semantic Web, statements that describe concepts, logic, restrictions, and

individuals. The statements share the same standards to enable sharing and integration, which take advantage of the semantics.

The Semantic Web is best understood in comparison to the World Wide Web (WWW). Table 1-1 compares the two. Rather than being a substitute for the WWW, the Semantic Web extends it through useable, standardized semantics that draw deeply on academic research in knowledge representation and logic to approach the goal of ubiquitous automated information sharing.

Table 1-1 Comparison of WWW and SW

FEATURE	WWW	SEMANTIC WEB
Fundamental component	Unstructured content	Formal statements
Primary audience	Humans	Applications
Links	Indicate location	Indicate location and meaning
Primary vocabulary	Formatting instructions	Semantics and logic
Logic	Informal/nonstandard	Description logic

The WWW consists primarily of content for human consumption. Content links to other content on the WWW via the Universal Resource Locator (URL). The URL relies on surrounding context (if any) to communicate the purpose of the link that it represents; usually the user infers the semantics. Web content typically contains formatting instructions for a nice presentation, again for human consumption. WWW content does not have any formal logical constructs. Correspondingly, the Semantic Web consists primarily of statements for application consumption. The statements link together via constructs that can form semantics, the meaning of the link. Thus, link semantics provide a defined *meaningful* path rather than a user-interpreted one. The statements may also contain logic that allows further interpretation and inference of the statements.

The flexibility and many types of Semantic Web statements allow the definition and organization of information to form rich expressions, simplify integration and sharing, enable inference, and allow meaningful information extractions *while* the information remains distributed, dynamic, and diverse. Simply put, the Semantic Web improves your application's ability to effectively utilize large amounts of diverse information on the scale of the WWW. This is accomplished through a structured, standardized approach for describing information so as to allow rich information operations.

Semantic relationships form the Semantic *Web*. The relationships include definitions, associations, aggregations, and restrictions. A graph helps visualize a collection of statements. Figure 1-3 shows a small graph of statements.

Statements and corresponding relationships establish both concepts (e.g., a Person has a birth date; note the double lines) and instances (e.g., John is a friend of Bill). Statements that define concepts and their relationships form an *ontology*. Statements that refer to individuals form instance data. Statements can be asserted or inferred. The former requires the application to create the statement directly, to assert the statement (solid lines). The latter requires a reasoner to infer additional statements logically (dashed lines). That John is an associate of Bill is inferred from the asserted statements. Future chapters cover these concepts in more detail.

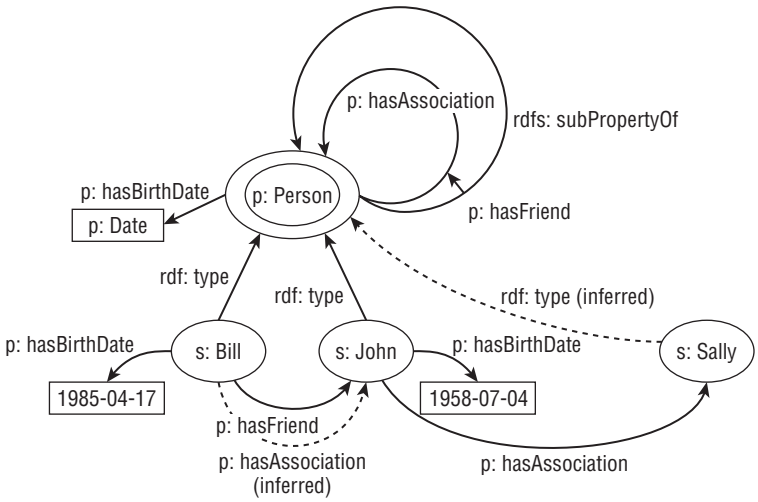

Figure 1-3 Example graph

Semantic Web statements employ a Semantic Web vocabulary and language to identify the different types of statements and relationships. Various tools and application frameworks use the statements through an interpretation of the vocabulary and language. Exploring and applying these tools and frameworks in relationship with the Semantic Web keywords is the focus of this book.

The Semantic Web offers several languages. Rather than have one language fit all information and programming needs, the Semantic Web offers a range from basic to complex. This provides Semantic Web applications with choices to balance their needs for performance, integration, and expressiveness.

A set of statements that contribute to the Semantic Web exists primarily in two forms; knowledgebases and files. Knowledgebases offer dynamic, extensible storage similar to relational databases. Files typically contain static statements. Table 1-2 compares relational databases and knowledgebases.

Table 1-2 Comparison of Relational Databases and KnowledgeBases

FEATURE	RELATIONAL DATABASE	KNOWLEDGEBASE
Structure	Schema	Ontology statements
Data	Rows	Instance statements
Administration language	DDL	Ontology statements
Query language	SQL	SPARQL
Relationships	Foreign keys	Multidimensional
Logic	External of database/triggers	Formal logic statements
Uniqueness	Key for table	URI

Relational databases depend on a schema for structure. A knowledgebase depends on ontology statements to establish structure. Relational databases are limited to one kind of relationship, the foreign key. The Semantic Web offers multidimensional relationships such as inheritance, part of, associated with, and many other types, including logical relationships and constraints. An important note is that the language used to form structure and the instances themselves is the same language in knowledgebases but quite different in relational databases. Relational databases offer a different language, Data Description Language (DDL), to establish the creation of the schema. In relational databases, adding a table or column is very different from adding a row. Knowledgebases really have no parallel because the regular statements define the structure or schema of the knowledgebase as well as individuals or instances. This has many advantages that we will explore in future chapters.

Take a look at the Semantic Web. Go to `http://www.geonames.org` and build a query. The application consists of many integrated information sources. The application is based on the ontology at `http://www.geonames.org/ontology`. Your Semantic Web application could also tap directly into this source and instantly gain access to this large, dynamic knowledgebase. These queries go well beyond simple tag or keyword searching and, therefore, provide a more focused extraction into a large information base.

One last area to consider is the Semantic Web's relationship with other technologies and approaches. The Semantic Web *complements* rather than *replaces* other information applications. It extends the existing WWW rather than competes with it. The Semantic Web offers powerful semantics that can enrich existing data sources, such as relational databases, web pages, and web services, or create new semantic data sources. All types of applications can benefit from the Semantic Web, including standalone desktop applications, mission-critical enterprise applications, and large-scale web

applications/services. The Semantic Web causes an evolution in the current Web to offer richer, more meaningful interactions with information. Our solutions throughout the book touch on these areas to illustrate the many ways the Semantic Web can enhance your software solutions.

Identifying the Major Programming Components

A Semantic Web application consists of several discrete components. Future chapters examine each one in detail and the programming examples make extensive use of each. First, we must define each one, note its purpose, and outline how the components contribute to form effective Semantic Web solutions. Some we have already introduced. They fall into two major categories: major Semantic Web components and the associated Semantic Web tools.

Figure 1-4 illustrates the major components surrounded by tools.

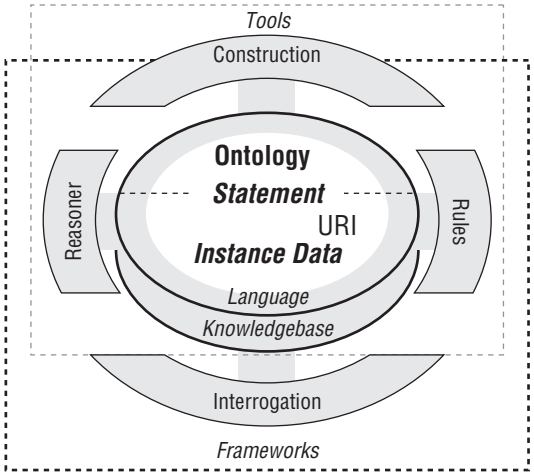

Figure 1-4 Major Semantic Web components

The core components consist of a Semantic Web statement, a Uniform Resource Identifier (URI), Semantic Web languages, an ontology, and instance data.

Statement: The statement forms the foundation of the Semantic Web. Each statement consists of multiple elements that typically form a *triple*. The triple consists of a subject, predicate, and object (e.g., John isType Person). The simplicity belies the aggregated complexity, as a solution combines thousands, even billions of these formal statements. Statements define information structure, specific

instances, and limits on that structure. Statements relate to one another to form the data web that constitutes the Semantic Web. The simple approach achieves powerful, flexible expressions.

URI: A Uniform Resource Identifier provides a *unique* name for items contained in a statement *across the entire Internet*. Thus, each component of a statement—subject, predicate, and object—contains a URI to affirm its identity throughout the entire WWW. This eliminates naming conflicts, ensures that two items are the same or not, and can also provide a path to additional information. A URI provides an expansive namespace—key to addressability regardless of scale. A URI could include a Uniform Resource Locator (URL), which may be dereferenced for useful additional information, or an abstract Uniform Resource Name (URN). Thus, the URI can also offer an accessible location contained within the URL. This extends to Internationalized Resource Identifiers (IRIs) covered in Chapter 3.

Language: Statements are expressed in accordance with a Semantic Web language. The language consists of a set of keywords that provide instruction to the various Semantic Web tools. In keeping with the variety and dynamics of the Internet, there are several languages for you to choose from. The languages offer various degrees of complexity and semantic expressiveness. Therefore your Semantic Web solutions can balance performance requirements and expressiveness. Higher levels of expressiveness often demand additional processing and storage resources. Future chapters cover all the terms contained in each language and their purposes.

Ontology: An ontology consists of statements that define concepts, relationships, and constraints. It is analogous to a database schema or an object-oriented class diagram. The ontology forms an information domain model. Many rich ontologies exist for incorporation into your applications. Your applications can use them directly or adapt them to your specific needs. An ontology may capture depth in areas such as finance and medicine, or capture breath in describing common objects, or present a hybrid of depth and breath. An effective ontology encourages communication across applications within the ontology's perspective. Of course, your Semantic Web solutions can create an ontology from scratch, but this isn't our recommendation. Instead, it is best when a Semantic Web application taps into the existing ontologies covering many domains. Using or augmenting an existing ontology leverages a well-thought-out and tested information domain and provides your solution with higher quality and greater development speed. Your added statements can focus on forming the ontology for your specific problem domain while leveraging ontologies from elsewhere.

Instance Data: Instance data is the statements containing information about specific instances rather than a generic concept. *John* is an instance, whereas *person* is a concept or class. This is analogous to objects/instances in an object-oriented program. Interestingly enough, instance data need not bind to the ontology (although in many cases this is quite useful). Instance data forms the bulk of the Semantic Web. An ontology containing the concept *person* may be used by millions of instances of *person*.

In order to exercise the Semantic Web, you need tools and frameworks. Tools come in four types: construction tools to build and evolve a Semantic Web application, interrogation tools to explore the Semantic Web, reasoners to add inference to the Semantic Web, and rules engines to expand the Semantic Web. Semantic frameworks package these tools into an integrated suite.

Construction tools: These tools allow you or your application to construct and integrate a Semantic Web through the creation or import of statements for the ontology and instances. Several GUI-based tools allow you to see and explore your data web to form a useful Semantic Web editor. Several programming suites outline an application-programming interface (API) to integrate with your program.

Interrogation tools: These tools navigate through the Semantic Web to return a requested response. There are various interrogation methods ranging from simple graph navigation, to search, to a full query language. Effective interrogation surfaces the usefulness of the Semantic Web.

Reasoners: Reasoners add inference to your Semantic Web. Inference creates logical additions that offer classification and realization. Classification populates the class structure, allowing concepts and relationships to relate properly to others, such as a *person* is a *living thing*, *father* is a *parent*, *married* is a type of *relationship*, or *married* is a *symmetric relationship*. Realization offers the same, for example, the *John H* instance is the same as the *J H* instance. There are several types of reasoners offering various levels of reasoning that future chapters explore. Reasoners often plug into the other tools and frameworks. Reasoners leverage asserted statements to create logically valid ancillary statements.

Rules engines: Rules engines support inference typically beyond what can be deduced from description logic. They add a powerful dimension to the knowledge constructs. Rules enable the merging ontologies and other larger logic tasks, including programming methods such as count and string searches. Rules engines are

driven by rules that can be considered part of the overall knowledge representation. Each rule engine adheres to a given rule language. Future chapters explore several of the available rules engines.

Semantic frameworks: These package the tools listed above to work as an integrated unit. Our book focuses on open-source alternatives for both a graphic integrated development environment (IDE) and an application programming interface (API). This allows you to get started programming immediately. There are also several excellent commercial semantic frameworks.

Statements, URIs, languages, ontologies, and instance data make up the Semantic Web, the connected semantic information. Semantic Web tools build, manipulate, interrogate, and enrich the Semantic Web. The book explores both in parallel with growing sophistication with each chapter.

Determining Impacts on Programming

In order for your applications to take full advantage of the Semantic Web and its tools, your applications must adapt to its expectations and impacts. We organize the programming impacts into four categories.

Web data–centric: Your Semantic Web application should place data at its center. Data is key.

Semantic data: Your Semantic Web application should place meaning directly within the data rather than within programming instructions or pushed out for user interpretation.

Data integration/sharing: Your Semantic Web application should attempt to access and share rich information resources throughout the WWW when appropriate, including taking advantage of the many existing data sources.

Dynamic data: Your Semantic Web application should enable dynamic, run-time changes to the structure and contents of your information.

These four impacts potentially change the way you design and program an application. They guide your solution to make optimal use of the Semantic Web. Figure 1-5 illustrates the four programming impacts.

Establishing a Web Data–Centric Perspective

Most applications are centered on programming instructions. They revolve around the program: if/then, while, for, int .... A Semantic Web application is just the opposite. It is all about the data. The richness of Semantic Web data lightens the programming burden. This decouples the data from the

programming instructions and produces a cleaner, more flexible solution. The programming instructions focus on programmatic chores while leaving complex information representation within the Semantic Web.

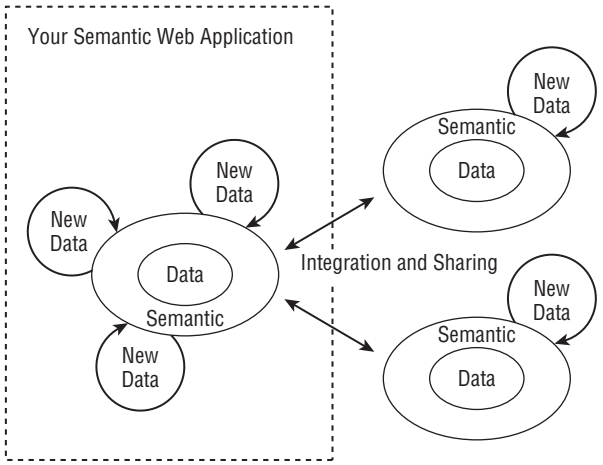

Figure 1-5 Four programming impacts of the Semantic Web

The Semantic Web application is web-centric; it takes advantage of the scale, diversity, and distribution found on the WWW. Many current applications struggle with these issues. They are unable to take full advantage of the WWW and thus remain trapped behind firewalls, serving in a limited, isolated capacity. The Semantic Web takes advantage of the size and diversity of WWW through the establishment of standard, expressive information. It was designed to take advantage of the quantity, diversity, and distribution found in the WWW.

Your programming perspective advances from a small, often-isolated, program-centered perspective (and in this case even enterprise computing could be considered small) to a global, interdependent, web-centered data perspective.

Expressing Semantic Data

The Semantic Web employs a set of new information standards, standards that others can share. As participation and adoption grow, your applications can quickly incorporate new, rich information sources. Standards in the WWW released useful content, mostly for humans. Standards applied in open-source software released powerful programs, many of which became application standards, such as the Apache Web Server. Now Semantic Web standards open up useful, rich information.

The rich standards in the Semantic Web extend well beyond syntax into forming a semantic standard. Syntax enables technical operations through the identification of the actual content. Syntax distinguishes data but not knowledge. Another part of the program—or often the end user—provides the meaning.

Syntax identifies special data items. The syntax of HTML identifies special data items called *tags*. Tags have proven helpful in many information-rich areas like photos and web pages. A single concept can have many tags. Tags can be ambiguous, and it is often difficult to discern what a specific tag means. Tags are often atomic and isolated; a *boat* tag is completely independent from a *yacht* tag. Without semantics, a boat tag and yacht tag reveal no similarities. The Semantic Web goes *beyond tags*. The Semantic Web connects these concepts through its web to improve the semantics and construct an expansive context for application consumption.

The Semantic Web enables *higher levels of information expressiveness*. Limits on information expressiveness challenge programming solutions. Variables, structures, relational tables, and so on all have their limits and peculiarities. Databases, for example, typically constrain the type of data (e.g., integer) but not its use (e.g., only on Fridays) or range (e.g., values between 5 and 9). Applications must absorb this lack of expressiveness through additional programming instructions. Thus, valuable knowledge is distributed haphazardly between data storage and programming instructions due to its inherent limitations. This often leads to brittle, inflexible code and misinterpretations, multiple interpretations, and errors. The Semantic Web offers extensive methods to define information, its relationships to other information, its conceptual heritage, and logical formation. This allows your program to capture more of its intelligence in one standardized way—the Semantic Web.

Relationships take on a primary role in the Semantic Web. In fact, they are the very fabric of the Semantic Web. Object-oriented solutions make relationships secondary to the objects themselves. Relationships do not exist outside of an object. Relationships are dependent on their associated object class. Relationships cannot be repurposed for other classes. Relationships in the Semantic Web exist distinct from the concepts that they join. Relationships are free to join any collection of statements. This allows relationships to have inheritance and restriction rules of their own. For example, a social network relationship within the Semantic Web could offer an `associatedWith` relationship that contains a subrelationship `ownsBy` and another subrelationship `friendOf`. Figure 1-6 illustrates an example graph of these relationships.

Due to the inheritance of the `associatedWith` relationship, an application could query for all `assocatedWith` data. This would include both people and, in this case, cars.

Similarly, statements that refer to instances, or instance statements, are also held in distinct regard. Instances are somewhat analogous to objects in an

object-oriented solution. An object in an object-oriented solution is dependent on its defined class. In fact, the object is defined as an instance of its associated class. The object's identity emerges directly from its classes. An object is bound for its lifetime to its class. The Semantic Web offers flexibility with instances. An instance is not permanently bound to any class or set of classes. In fact, an instance can have no class at all and merely stand alone as an instance statement or be associated with multiple classes. This allows the application to add instances before it understands their connections to classes. Your application can dynamically change the association of an instance with its class. Your application can also assign multiple classes to a given instance. This allows the flexibility to form and capture information independent from class definitions. These assignments of instances to class can occur at any time.

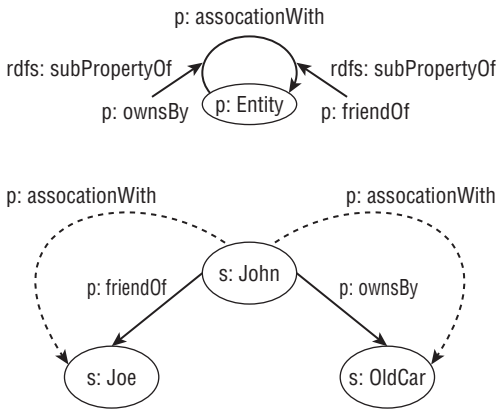

Figure 1-6 Semantic Web relationships

Fundamentally, the Semantic Web offers a new way to describe and share information, a description that flexibly contains and reveals its semantics.

Sharing Data

The ability to exchange information greatly increases the information's value. Easy information exchanges allow the applications to use the best, most up-to-date information rather than forming the information from scratch. Easy exchange also leads to real reuse.

The Semantic Web enables semantic *machine readability* for exchanging information between applications. Semantic machine readability goes beyond the mere exchange of bits to exchange meaning. This is especially important for large, distributed data exchanges typical of the WWW. Large data integration belies significant challenges in data friction and data failures. These challenges

represent the part of the *integration iceberg* that is below the surface, large and hidden from the original integration task. Similar to an iceberg's hidden mass, these challenges often reveal themselves at stressful, challenging times.

Data friction results from the misalignment of the data due to different technologies and data models. For instance, an Oracle database has a table that refers to a *Person*, and a MySQL database has a corresponding table that refers to an *Individual*. One information source is centered on *work organizations*; another application is centered on *members*. Each data source maintains its own perspective based on its needs. Neither is wrong, but integrating them creates friction that's usually ameliorated through extensive and complex translation programming code. Semantic Web solutions reduce this friction through semantics that relate similar concepts using Semantic Web statements. This externalizes the friction, allowing the Semantic Web to directly address it.

Data failures result from missing data, conflicting data, and incorrect data. Large integrations are full of data failures. The semantics and logic structure that form the Semantic Web add missing data as well as the identification and sometimes correction of data errors and data conflicts. The flexibility of the Semantic Web allows these corrections to occur continuously as the data is integrated or modified. Continuous correction allows the resolution of data failures to occur gracefully throughout the data's life. There is no need for one big data cleanup activity usually demanded in other data approaches. The Semantic Web produces more useful, shareable, and up-to-date data resources. It pragmatically deals with the realities of large-scale data integration.

Naming represents a challenge to sharing information. A name identifies the information and possibly its location. Many computer resources, such as data in a database, have a name, a table name, a variable name, or the like. A useful name distinguishes it from other related resources. The related resources occupy a *namespace* where all names promise uniqueness. In order to leverage the WWW namespace, a resource must establish uniqueness across the entire WWW or else face limitations with related resources. Many namespaces limit scope even within the WWW. For example, AOL buddies maintain a namespace distinct from Yahoo buddies. Each buddy remains unique only to its associated Instant Messaging service. Semantic Web artifacts are *inherently* unique across the full span of the Internet. In addition, names are even more useful if they provide information as to the resource's location. A URL provides both a unique name and a location (along with the possibility of additional information). Elements within the Semantic Web are uniquely identified with Uniform Resource Identifiers (URIs), which can also contain a resolvable location, if so desired.

Machine readability, data friction and data failure management, and unique, resolvable names foster a much smoother path to large-scale data integration and sharing than other approaches.

Making Data Dynamic and Flexible

Dynamic change is a tenet of the WWW. It is continuous and often unexpected. Yet many applications are frozen and trapped by their initial requirements. This is especially true with data and its representations. Many data representations are, therefore, perpetually designed to address yesterday's requirements. Freeing that data from traditional approaches is possible, but it comes at a high price, increased complexity. A typical workaround to deal with change reuses database strings for new and sometimes multiple purposes. One application uses the database string field for status information for its customers in the database. Another application uses the same string field for pending adjustments to its customers. Sooner or later the two incompatible approaches collide or surrender to the increasing complexity. The technique works for a short time but eventually implodes. Properly designed Semantic Web applications allow the inclusion of new data at any time. This allows the Semantic Web application to gracefully align with current needs and avoids the traditional workarounds found in brittle designs.

The Semantic Web enables information to *figure things out* through inference. Imagine information that adds information to itself, by itself. This is a power that few information technologies offer. Once information has been properly described, a reasoner can infer new relationships. For example, say your information contains the following two facts; *John is a male, a male is a type of person*. The system could infer that *John is a person*. Thus, a query requesting all persons would return *John*. Of course, this is a simple, straightforward example, but it serves to illustrate one of the principal benefits of encoding the semantics of data along with the raw values themselves. A reasoner can also identify logical contradictions within your information. For instance, if you made the claim that men and women are mutually exclusive groups, and somewhere was the datum that John is a female, then the reasoner could use the information from above to recognize contradictory data. Of course, this is obvious and simple, but as the statements get more numerous, complex contradictions are much more difficult to identify.

Inference makes each data item more valuable, because it can have an effect on the creation of new information. This can be both positive and negative. Each new piece of information has the capacity to add a great deal of new information via inference, but that means that extra care must be taken to validate new information. The inference process, just like all computer systems, is vulnerable to the ''garbage in, garbage out'' phenomenon, but inference amplifies this concern.

The Semantic Web provides meaningful links between related bits of information, a new form of *information navigation*. This semantic metadata allows a new type of information discovery and flexibility. You could start at one point in the Semantic Web and explore it following a particular set of

relationships. For example, you could start at information about a friend, then follow the relationship to her friends to find out their interests. One friend of your friend might have an interest new to you, so you could follow a link to a definition of the interest. Maybe it sounds interesting, and you've found a new hobby—and possibly a new friend, too! Naturally you can follow links on web pages to do this already. But the Semantic Web offers the possibility of having a program do it just as easily. Traditional searching and querying serve as a nice complement to this sort of semantic navigation.

The Semantic Web is dynamic. An application can add information at any point. Concepts can evolve at any time to become more useful or precise. The WWW is a lively place, and the Semantic Web provides technologies to leverage that change rather than avoid it. When amplified by the power of inference, one new statement could ripple through the knowledgebase to transform it into a more useful form. Similarly, that statement could be removed, and the information would be back to its original state.

Finally, the Semantic Web is inclusive. The Semantic Web's ability to express information flexibly makes it ideal to incorporate other forms of data. The Semantic Web can include semantic translation rules and statements allowing the incorporation of other types of data from relational databases, XML web services, and even simple comma-separated lists. Future chapters illustrate each of these techniques. Therefore, these Semantic Web techniques can unify and enrich other information sources and services. This is vital capability, because much of the valuable information on the Internet, in the enterprise, or on your laptop is *not* currently part of the Semantic Web.

The Semantic Web impacts require a new perspective in developing applications, a perspective based on a data-centric model leveraging large amounts of diverse, distributed data that contains greater expressiveness, easier sharing, and greater flexibility.

Avoiding the Roadblocks, Myths, and Hype

The new programming perspective must overcome the various misunderstandings already present in the nascent Semantic Web. Roadblocks, myths, and hype cloud your ability to move straight into programming, sometimes stopping you altogether. If not affecting you, they may block the way for your team or organization to take advantage of the Semantic Web. Sometimes the largest hurdles are the ones we make up.

Semantic Web Roadblocks

Roadblocks can stop programming efforts dead in their tracks before they gain momentum. We deal with three: web-centric development; taking comfort in dirty, conflicting data; and dealing with the dynamic addition of new data.

Technically the Semantic Web demands a new perspective on information. Think how relational database technology creates a *data perspective*, a perspective based on the low-level details of tables, columns, rows, and keys. Now you must establish a Semantic Web data perspective. This perspective has two parts: data-centric programming and distributed information programming.

We previously covered web data–centric impact, but unfortunately it can also become a roadblock to using the Semantic Web effectively.

In addition, information is never all in one place. Traditional approaches seek to provide integration via the Extract, Transform, and Load (ETL) approach. This involves a laborious process of acquiring the information, transforming it, and finally integrating it into some single, unified store. This approach has several problems: it does not scale; it incurs latency, leading to incorrect information; it is error prone; it doesn't handle change well; and it has a single point of failure. The small scope of most traditional applications reduces the seriousness of these challenges, but the applications remain constrained by them. Integrating via Semantic Web programming means creating an interface with the real source and not usually copying it. Real-time semantic translations allow your solution to scale while maintaining up-to-date information. Accordingly, Semantic Web solutions must also deal with performance issues and resource failures due to the distributed nature of the Semantic Web.

The second roadblock is the failure to recognize dirty, noisy data as an advantage. The WWW is full of data, lots of it. Some of it is useful, some of it is not, and some of it is just plain wrong. Traditional solutions that attempt to incorporate such data try to *fix* and *correct* the data prior to using it. Subscribing programs assume the data is clean and correct. Again, a small application may actually correct all its data, but that is simply not a possibility for the global information space of the WWW. Instead, large data integration produces dirty, noisy information with apparent conflicts, duplication, and errors. In order to *fix it*, your program would first have to know the truth. Unfortunately, often in the WWW there is no ground truth, no overriding authority. Rather than mount the quixotic struggle to right all the wrongs, a Semantic Web application can operate effectively within this global mess. A Semantic Web application creates concepts that it considers useful for its needs. These concepts can incorporate dirty information or not. For example, a Semantic Web application could collect vendor feedback scores from several sites and create a *reputation* concept for vendors. The concept can choose to include information or not, depending on your concerns and values for a given situation. Conflicting information could be incorporated into the concept, possibly lowering the confidence in the reputation. Likewise, supporting information could increase the confidence. The flexibility of the semantics enables applications to define reputation in different ways and manage conflicting information. This flexibility allows an application to incorporate all or some of the relevant information and map it to its specific concept. Traditional applications fail to offer such flexibility or take

advantage of this flexibility. Similarly, a Semantic Web application can handle missing data or incorrect data, as you shall see when we cover reasoning and related topics.

The dynamic flexibility of the Semantic Web can also form a roadblock. A Semantic Web application is not nailed down like traditional applications. The dynamic nature requires your application to manage change. Your Semantic Web application must remain open to possibilities, at any time. Remaining open is not a perspective easily gained. A Semantic Web application is designed to handle new, incoming information. As mentioned previously, this information could be conflicting or contain errors. A Semantic Web application maintains a nimble, agile view to data. It is not as fixed and controlled as traditional database applications. Many upcoming examples will help to provide a more concrete view of this important perspective.

So beware of these roadblocks: programming instructions over the creation of statements for the Semantic Web, trying to gather all the data into one place, and fearing the dynamics that the Semantic Web allows. Heeding these roadblocks causes Semantic Web solutions that mimic the constraints of traditional systems and thereby fail to gain the full advantages of the Semantic Web.

Semantic Web Myths

Several myths can also impede your efforts. They were founded in the development of traditional systems: developing one data model, developing one data view, and the inability to accept change à la the human acceptance myth.

Building the *one* of anything has been the scourge of technical progress. Have you been involved in building the *one* corporate database, the *one* enterprise framework, the *one* perfect web service? One of the biggest myths of the Semantic Web and one sure to undermine its success is the pursuit of the one big information model. We have seen several efforts stumble over this unachievable challenge with lengthy debates (*is a thing an object*?). Do not fall into this trap. It is the black hole of the Semantic Web, where effort goes in and nothing ever comes out. Even in some other universe where it works, it ultimately won't scale anyway. The Semantic Web is designed to support a multitude of distributed information sources with a multitude of perspectives. Your solutions need to maintain this perspective; *you should too*. The Semantic Web mirrors the WWW itself, so a Semantic Web application seeks to leverage the Web's vast distributed information. This small paragraph will not stop the insanity of others from pursuing the one perfect model, but hopefully it will stop you. If you hear someone discussing the one big data model, run!

The next myth, which aligns perfectly with the one model, is the one view. The Semantic Web allows a solution to provide deep customized views into information. This freedom is at the very heart of the Semantic Web. Rather than be a straightjacket of control, an effective Semantic Web application allows

any view for information analysis. Accordingly, this multiple-information perspective aligns perfectly with agile methods. Even in a limited-scope information source, it is unlikely that your initial design will work perfectly. The same ability to absorb other information sources easily works to absorb your future changes. This also stops the initial paralyzing steps of nailing the perfect, albeit limited, information model. This perspective allows you to get started quickly and then adapt and evolve as the development unfolds. Accordingly, this also encourages a decoupled, modular design. Multiple views also help manage the dirty-data phenomena. What is one person's data dirt may provide someone else with a useful insight.

In addition, both the one information model and the one view suffer from a further myth, Not Invented Here (NIH). The Semantic Web encourages just the opposite. A Semantic Web solution should first look for existing semantic sources and then add customizations. Only the last resort creates an information source from scratch.

Finally, the acceptance myth: Change is difficult, and we have never experienced a Semantic Web implementation that was instantly accepted. New technologies, especially those dealing with sensitive data, scare people, as they should. Change also scares people. You need to address this up front and prepare for resistance. As with all new technologies and approaches, there is risk; however, risk works both ways: positive and negative. Positive risk may provide your application with a real advantage compared with traditional approaches. Negative risk may saddle your application with extra training and support costs. Simply play up the positives by taking advantage of the Semantic Web and work to mitigate the disadvantages. Face this directly and honestly to maximize the benefits while realizing that nothing is a silver bullet.

Semantic Web Hype

Hype dooms a technology, for it overpromises (lies), and in doing so, it disappoints, just as we need to invest and learn it. As with any major technology, the Semantic Web is full of hype, some positive and some negative. Much of the hype comes from its artificial intelligence (AI) roots. In the 1980s, AI had everyone excited about the friendly computer agent that would achieve a level of sentience. Videos demonstrated computers of the future that offered human insights and capabilities. This, of course, didn't happen and produced a cynical view of powerful AI research. AI had stunning successes, but much of this was buried under the disappointment caused by the hype. Adding semantics through relationships and logic does not achieve AI. The Semantic Web did, however, incorporate excellent AI research of the past decades. The Semantic Web offers a useful improvement in leveraging information. It is an evolutionary step in making our information work harder for us. In ways, the Semantic Web helps fulfill some of the AI goals by providing a rich, expressive

data form that is machine readable, but the Semantic Web will not achieve sentience anytime soon.

Hype can also make a complex technology seem too simple. Tough, challenging problems demand complex technical solutions. Would you want your MRI performed by a hacked Visual Basic script? Complexity is valuable as long as it remains focused on the problem. Unnecessary complexity in the solution makes the problem's complexity that much more difficult. Modern computer languages have removed much of the complexity of the underlying computer resources so that applications can step up to more complex solutions. Similarly, the Semantic Web has removed much of the complexity of forming and exchanging complex information. It has not removed the complexity of the information itself but rather enabled its capture. Rocket science is still rocket science, but now information about it can be more easily expressed, shared, and reasoned over. The Semantic Web reduces unnecessary complexity to focus on the necessary complexity, that of managing all the information and knowledge we produce.

Finally, hype can overpromise the inherent challenges in using a new technology. Our efforts in the real world took us into areas of the Semantic Web that have never been applied to real solutions. We braved this world so that you don't have to. That doesn't mean that it is all finished. Much remains to be done, and it is our hope that you will contribute to its advancement. The Semantic Web is improving with large investments across the globe, but it is not without its growing pains. That is simply the truth in working with any new, emerging, and evolving technology. Do not expect a perfect world with respect to tools, frameworks, and the semantic language itself. You may hit some bumps, but hang in there; it will be worth it.

Understanding Semantic Web Origins

The origin of the Semantic Web comes from the quest to externalize knowledge. Despite a long history of advancements, the quest remains for us today. We have tried in so many ways and yet remain frustrated with our efforts. Read *Gödel, Escher, Bach: An Eternal Golden Braid* by Douglas R. Hofstadter (Basic Books, 1999) if you want some reassurance that the human brain processes information in ways we simply don't understand. It is no surprise we have trouble translating information to a machine. The computer demands exactness and precision, two areas that humans often fudge. Herein lies the dilemma: Computers can do useful things only if we explain those things in excruciating, precise, and consistent detail. This struggle has produced many types of data, information, and even knowledge formats. Commonly we can refer to them as forms of knowledge representation. They include network databases, relational databases, tree structures, objects, messages, and the like. Along

this path, we collectively have learned what works and what doesn't work for different problems. Often it is not an absolute answer but a contextual one. The Semantic Web is no different. It does not so much replace these approaches as give you a new approach. It embraces a new form of knowledge representation, a knowledge representation that leverages and improves on some previous methods.

The Semantic Web gained wide visibility following an article in *Scientific American* by Tim Berners-Lee, James Hendler, and Ora Lassila (`http://www.sciam.com/article.cfm?id=the-semantic-web`). The article outlined a pragmatic vision to improve the value of the information contained in the World Wide Web. It foretold many of the characteristics noted previously: machine readability, easy information integration, information inference, unique naming, and rich representations, among others. This led the authors to make a bold statement: ''The Semantic Web can assist the evolution of human knowledge as a whole.'' Bold but possible, given the current extent of information on the World Wide Web. We know that many answers to our questions are out there, but due to the current human-only readable form of the data, they remain only a tantalizing possibility. How many web pages can you digest to form an answer? How many do you need to read to gather all the little pieces that together form the answer you seek?

The Semantic Web did not emerge out of just one seminal paper, but many. The Semantic Web is based on sound, proven information techniques built on centuries of scholarly contributions. Two major disciplines contributed to the Semantic Web: graph theory and description logic.

Graph theory is at the heart of the Semantic Web. A graph represents nodes and relationships. Stepping back from the WWW, you can recognize a graph-like structure or web through its many hyperlinks. Graph origins predate, by quite a bit, the World Wide Web and even computer technology more generally. Graph theory started with a seminal paper from 1736: ''The Seven Bridges of Königsberg,'' by Leonhard Euler, the famous mathematician. This paper provided a solution to a basic question. Is there a route that allows the traveler to cross each of the seven bridges of Königsberg once and return to the starting point? The answer is *no*, and he solved it using a new branch of math, graph theory. With graph theory, he could *prove* that it is not possible. His graphs imply the answer. The only other approach is brute force—*try each path*. This is not only a tedious, exhausting method, but it leaves the questioner with troubling doubts, for it is possible that an untried path does exist. The search for a traveling route forms the foundation of establishing routes to useful information on the Web. Semantic Web solutions tap directly into the mathematical advantages that graph theory offers. This leads to data processing efficiencies.

Description logic also goes back, but not quite as far. Several papers outline the basics in the 1980s. Description logic holds the rules to construct

valid, useful knowledge representations, knowledge representations that are decidable, representations that can actually produce an answer. Undecidable representations result in representations that wrap around themselves, never reaching any conclusions. It derives from first-order logic. It defines expressions that are, well, *logical*, a formalism for representing knowledge (see *An Introduction to Description Logics* by Daniele Nardi and Ronald J. Brachman (Cambridge University Press, 2002). Description logic resulted from years of AI research aimed at capturing rich knowledge in an explicit, externalized form. Description logic makes information explicit rather than tacit. Tacit information is what gets diffused in traditional applications via if/else, special values, and shortcuts, because only humans really understand it (and we have trouble articulating it). The externalized form reveals the information for verification, integration, reasoning, and interrogation. Description logic includes many types of relationships beyond inheritance to provide the flexibility to form rich, complex concepts that maintain defendable logic. These external forms researched and proved in description logic translate into Semantic Web constructs. Thus, your Semantic Web programs can now take full advantage of description logic. Your Semantic Web application can tap directly into this powerful expression of information.

Why mention this history? Good ideas are good ideas, ideas that take many forms over the ages. The Semantic Web is the latest manifestation of graph theory and description logic. Your Semantic Web applications form instances of these advancements. Your programming efforts in the Semantic Web spring from this solid theoretical ground. You can defend your pursuit and your solutions. How many others could refer to a work from 1736 and exercise useful nuggets from AI? Semantic Web solutions incorporate advanced information theory. The pursuit of paths through bridges and information itself led to the new technologies that underlie the Semantic Web.

The Semantic Web is built not only on mathematical theories but also on fundamental Internet technologies and philosophies. The success of the WWW has taught us not to go it alone, at least if a technology wants to survive. Build success on other proven successes. The Semantic Web is no different. The Semantic Web supports the inclusive and evolutionary nature of the WWW. The Semantic Web layer cake illustration in Figure 1-7 demonstrates some key dependencies such as URLs and XML that form the foundation of the Semantic Web.

Figure 1-7 also shows that future Semantic Web capabilities will deal with trust and providence. Semantic Web applications also utilize Internet services such as DNS and even traditional relational database technologies found in implementations such as Oracle and MySQL. More fundamental than that, the Semantic Web depends on existing information sources, the more semantic the better. Successful Semantic Web solutions reach out to many diverse sources to fulfill their larger information need. This fits with the recently formed Web 2.0 philosophy, a philosophy based on extensive integration and user

contributions. Examples include Google maps, MySpace.com, Flickr.com, and Facebook.com. Semantic Web solutions make the most of available technologies. In addition, like any Web 2.0 technology, the Semantic Web benefits with each new additional application and information source. It benefits from the virtuous cycle of wide-scale contributions that established Web 2.0.

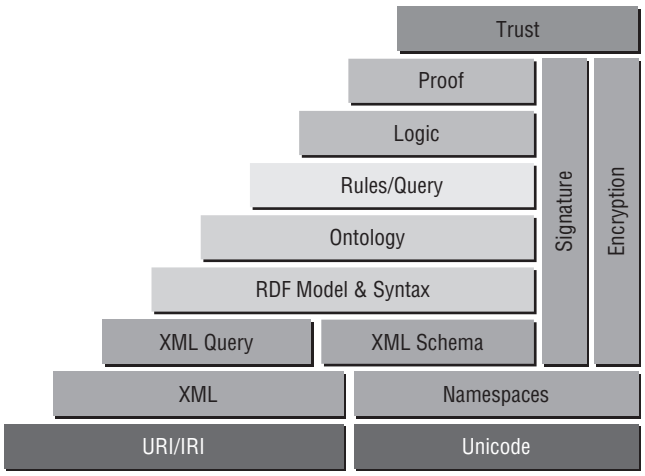

Figure 1-7 Semantic Web layer cake

The origin of the Semantic Web begins with our pursuit of externalizing what we know. It benefited from key advancements in graph theory and description logic to produce a viable knowledge representation that values existing technologies, applications, and data sources.

Exploring Semantic Web Examples

Now we'll demonstrate some actual Semantic Web information sources and applications. The following examples highlight some of the advantages of the Semantic Web applications. The examples include a variety in order to highlight key semantic features.

Semantic Wikis (*semantic-mediawiki.org*)

Wikis collaboratively build useful websites; however, the easy creation of content, web pages, and links belies the difficulty of fully leveraging the content. Wikis depend heavily on human-entered links. Content becomes

stranded or just plain lost if the link doesn't exist on an obvious page. The links themselves are almost nonsensical due to their automatic, dynamic creation. Semantic wikis allow the collaborators to enter semantics. This allows a user to query the wiki or semantically navigate to find information rather than just depending on the native links. In addition, applications can query the content and reuse it. This is vital as the wiki grows beyond a few pages. Since the wiki participates in the Semantic Web, inference and other tools can add to its value. The contributors enter semantic properties, and the associated values are bound to the entered content. The Semantic wiki can export the semantics to a file or an external application. This book was developed using a semantic wiki for collaboration and discussion. Figure 1-8 illustrates a basic query to find content.

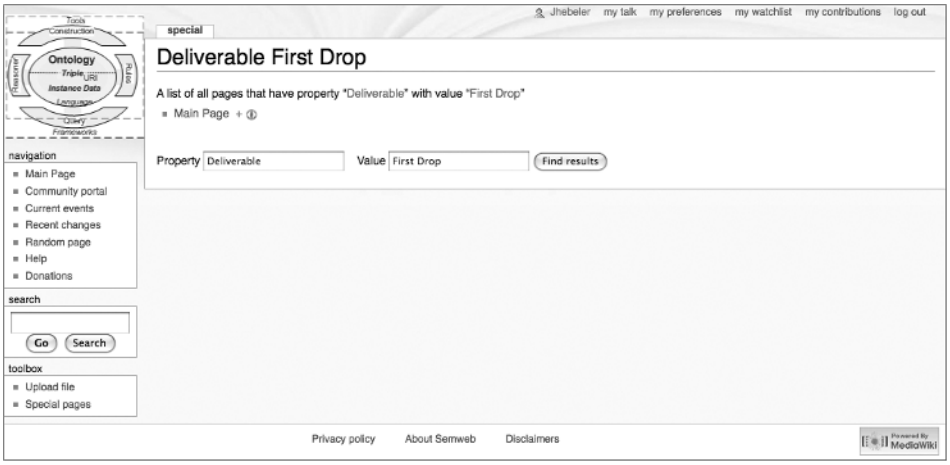

Figure 1-8 Semantic wiki

There are several rapidly evolving semantic wikis. They hold the promise to manage and leverage large amounts of user created content.

Twine (*www.twine.com*)

Twine is all about relationships, information relationships. From these relationships, other relationships, such as social relationships, emerge. The result is a *knowledge network*. It employs the Semantic Web to construct and manage many of these relationships. You can easily add information and meta-information from all types of sources, such as RSS feeds, emails, blogs, and direct file downloads. Figure 1-9 illustrates a twine about the ontologies.

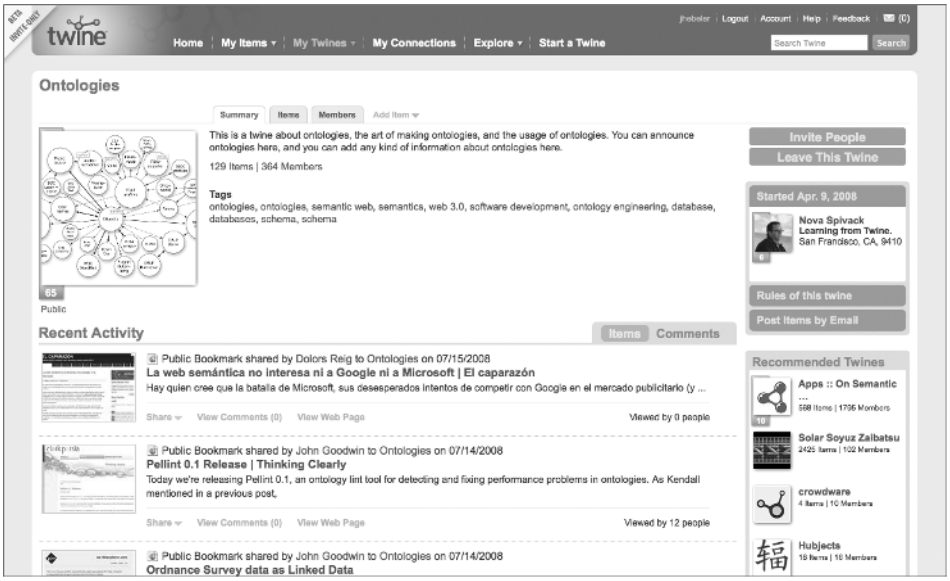

Figure 1-9 Twine relationships

Figure 1-10 illustrates adding information to an existing twine, in this case the BBN Semantic Web twine.

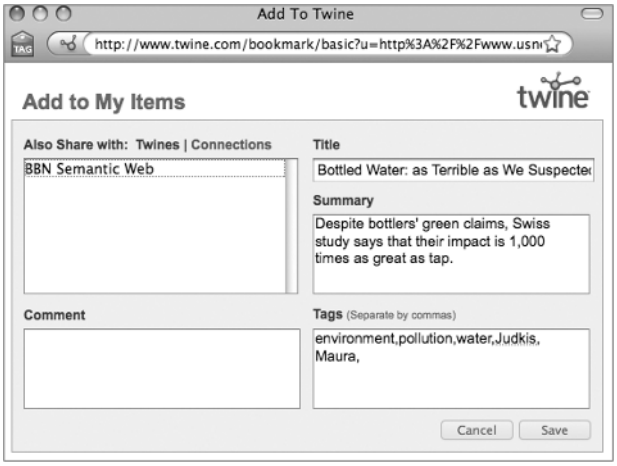

Figure 1-10 Contributing to a twine

The FOAF Project (*www.foaf-project.org*)

FOAF is not so much an application as an ontology used by many applications, including major ones discussed later in this book. The Friend of a Friend (FOAF) project was one of the first to recognize the simple power of social networks. The FOAF project offers tools to relate people through a model that contains typical social attributes such as a name, email address, interests, and the like. Tools allow you to create a model describing yourself (see `www.ldodds.com/foaf/foaf-a-matic`). The following code below is a FOAF file in TURTLE format (more on that later) describing some fictitious folks we explore in the next chapter. It contains basic information and their relationships to each other. Believe it or not, this basic start is a solid foundation for forming rich social networks. Future sections expand on this concept, including the next chapter that builds a Hello World semantic application.

```
<http://org.semwebprogramming/chapter2/people>
      rdf:type foaf:PersonalProfileDocument ;
      admin:errorReportsTo
              <mailto:leigh@ldodds.com> ;
      admin:generatorAgent
              <http://www.ldodds.com/foaf/foaf-a-matic> ;
      foaf:maker <http://org.semwebprogramming/chapter2/people#me> ;
     foaf:primaryTopic <http://org.semwebprogramming/chapter2/people#me> .


<http://org.semwebprogramming/chapter2/people#me>
      rdf:type foaf:Person ;
      foaf:depiction <http://semwebprogramming.org/semweb.jpg> ;
      foaf:family_name "Web" ;
      foaf:givenname "Semantic" ;
      foaf:homepage <http://semwebprogramming.org> ;
      foaf:knows
<http://org.semwebprogramming/chapter2/people#Reasoner> ,
<http://org.semwebprogramming/chapter2/people#Statement> ,
<http://org.semwebprogramming/chapter2/people#Ontology> ;
      foaf:mbox <mailto:dataweb@gmail.com> ;
      foaf:name "Semantic Web" ;
      foaf:nick "Webby" ;
      foaf:phone <tel:410-679-8999> ;
      foaf:schoolHomepage <http://www.web.edu> ;
      foaf:title "Dr" ;
      foaf:workInfoHomepage
              <http://semwebprogramming.com/dataweb.html> ;
```

```
      foaf:workplaceHomepage
              <http://semwebprogramming.com> .

<http://org.semwebprogramming/chapter2/people#Reasoner>
      rdf:type foaf:Person ;
      rdfs:seeAlso <http://reasoner.com> ;
      foaf:mbox <mailto:reason@firefox.com> ;
      foaf:name "Ican Reason" .
<http://org.semwebprogramming/chapter2/people#Statement>
      rdf:type foaf:Person ;
      rdfs:seeAlso <http://statement.com> ;
      foaf:mbox <mailto:mstatement@adobe.com> ;
      foaf:name "Makea Statement" .

<http://org.semwebprogramming/chapter2/people#Ontology>
      rdf:type foaf:Person ;
      rdfs:seeAlso <http://ont.com> ;
      foaf:mbox <mailto:ont@gmail.com> ;
      foaf:name "I. M. Ontology" .
```

The FOAF project encourages everyone to create and publish his or her FOAF model. Model readers can then recognize and incorporate this information into a contact list or a search response. Many existing web pages contain FOAF information.

RDFa and Microformats

Similar to FOAF, microformats and RDFa are also formats employed by many applications. These enter semantics directly into a typical XHTML web page. This demonstrates description data sharing using semantic formats. Tools integrated into the browser recognize these semantics and offer capabilities to use the semantics information. Although a small step, it is a step in the right direction. The lack of semantics on a basic web page makes search much more difficult. Microformats are a big improvement over screen scraping, which is a very brittle approach to obtaining information. A microformat could describe a contact as including a FOAF model, an event, resumes, job offerings, and much more. There are dozens of standard microformats, each for different types of information. See `www.microformats.org` for an extensive list. The Firefox extension, Semantic Radar, inspects web pages for semantic content and indicates this content via icons on the Firefox status bar. The web page illustration, Figure 1-11, contains several icons in

the lower right that provide links to the underlying RDF files, including a FOAF description. The second illustration, Figure 1-12, is the Ontology Online website.

Figure 1-11 Semantic Radar microformat plug-in

Semantic Radar provides a path to the semantic information. A partnering application could take this information and populate your calendar, your contact list, or your social networks. There is no easy way to accomplish this simple knowledge transfer without an agreed-upon semantics.

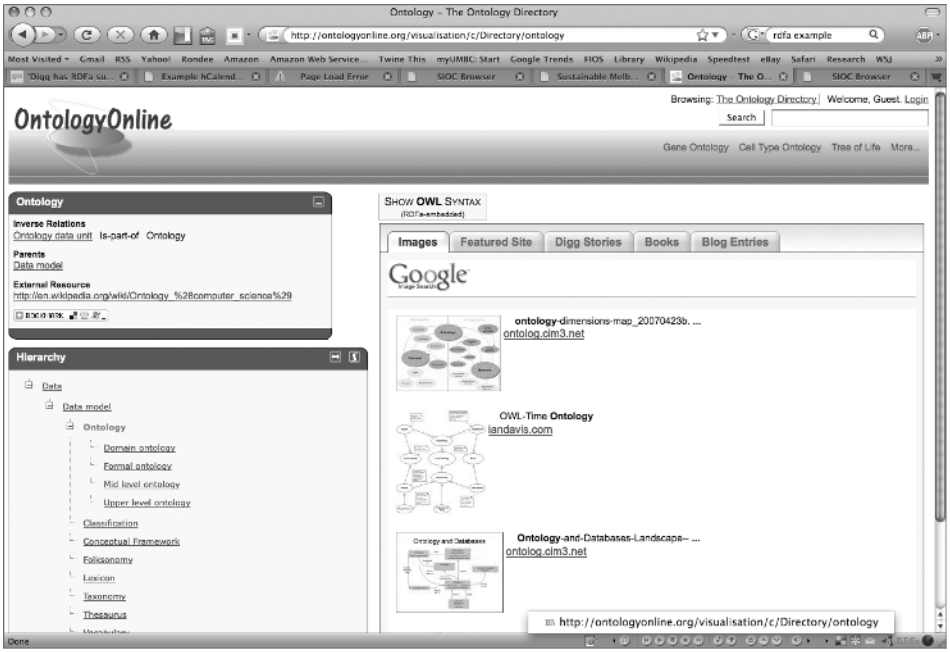

Figure 1-12 Semantic Radar example

Semantic Query Endpoint (*dbpedia.org/sparql*)

A semantic query endpoint offers the ultimate information exposure: a URL that answers your questions formed from a standard semantic query language, SPARQL. We focus on semantic query languages with SPARQL (SPARQL Protocol and RDF Query Language) in Chapter 7, ''Adding Rules'' but just note that the full power of that query can be directly exposed via the Web. This forms a Semantic Web offering. Without all the background, you can still give it try (`dbpedia.org/sparql`). Figure 1-13 illustrates a query response.

Semantic Search (*www.trueknowledge.com*)

Search is made much more powerful with the addition of semantics. Although still in beta, True Knowledge moves into this space. Others include www.opencalais.com and www.powerset.com. Figure 1-14 demonstrates the power of semantic search.

Some semantic wikis also offer up their content via a SPARQL endpoint.

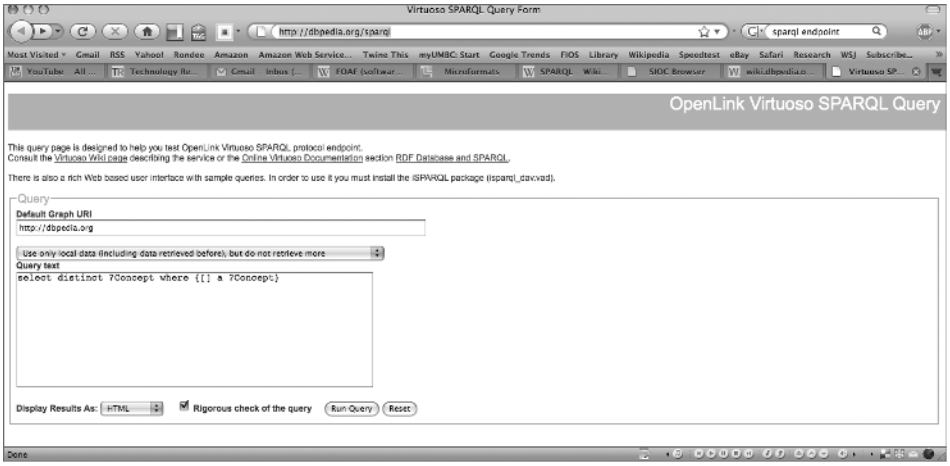

Figure 1-13 dppedia–SPARQL endpoint

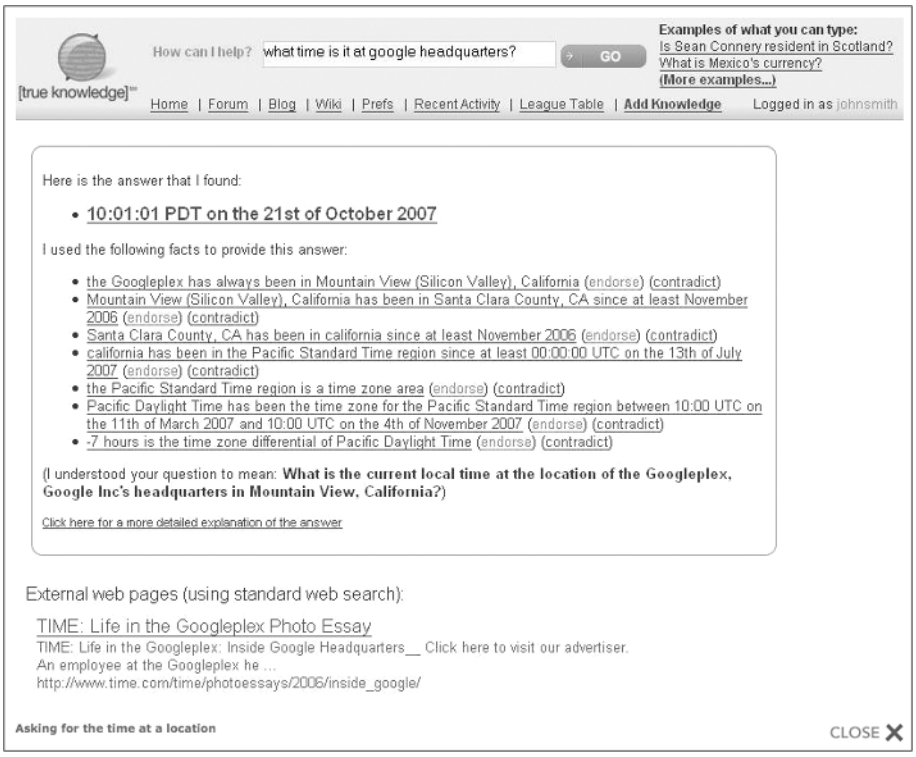

Figure 1-14 True Knowledge response

Okay, that gives you a taste of the possibilities ahead. On to forming your first Semantic Web application, the Hello Semantic World Tour, in the next chapter.

Summary and Onward

Your Semantic Web conceptual foundation is now established. The Semantic Web offers a new, more powerful way to create and share information. Its logical and standardized definition enables advanced information processing such as inference and validation. The Semantic Web requires a new programming perspective, one that turns distributed, confusing, and massive information into real solutions.

The Semantic Web has its fair share of roadblocks, myths, and hype. Seeing past these helps ensure a positive programming experience with a successful outcome. The Semantic Web is solidly grounded on graph theory and description logic. It provides a knowledge representation that is defendable and worthy of your investment.

Semantic Web programming consists of core components: statements, the URI, an ontology, and instance data managed and formed through the various construction tools, interrogation tools, reasoners, and rules.

Now, get programming—say hello to the Semantic Web Tour.

Notes

1. W3C Semantic Web Activity home page, http://www.w3.org/2001/sw/, which is ground zero for the Semantic Web URLs
2. ''Expressiveness and Tractability in Knowledge Representation and Reasoning,'' Hector J. Levesque and Ronald J. Brachman
3. ''Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering,'' eds. Phil Tetlow and Jeff Z. Pan (W3C Working Group, 2003)
4. An Introduction to Description Logics, Daniele Nardi and Ronald J. Brachman (Cambridge University Press, 2002)
5. ''Ontology-Driven Software Development in the Context of Semantic Web: An Example Scenario with Protégé/OWL,'' Holger Knublach
6. ''An Introduction to Model Driven Architecture,'' Alan Brown (IBM, 2004)
7. ''A Semantic Web Primer for Object-Oriented Software Developers'' (W3C Working Group, 2006)