

The Resource Description Framework

The Resource Description Framework (RDF) is an XML application for encoding metadata. In particular, it's well-suited for describing Web sites and pages so that search engines can not only index them, but also understand what they're indexing. Once RDF and standard RDF vocabularies become prevalent on the Web, searching can become finding. This chapter discusses the RDF statements about resources, basic and abbreviated RDF syntax, the use of containers to group property values together, and RDF schemas.

What Is RDF?

Metadata is data about data, information about information. For example, the text of a book is its data. The name of the author, address of the publisher, copyright date, and so forth is metadata about the book. Metadata has many uses on the Web, including organizing, searching, filtering, and personalizing Web sites. Accurate metadata should make it much easier to find the Web sites you want while ignoring the Web sites you don't want.

In order for metadata to have these benefits, however, Web sites, search engines, and directories must agree to use a standard format for metadata. The Resource Description Framework is a W3C-recommended XML application for encoding, exchanging, and reusing structured metadata. RDF vocabularies can describe rating systems, site maps, privacy preferences, collaborative services, licensing restrictions, and more.



In This Chapter

What is RDF?

RDF statements

Basic RDF syntax

Abbreviated RDF syntax

Containers

RDF schemas



In general, metadata vocabularies must be customized for each individual knowledge domain. However, RDF strives to create a convention that controls how the semantics, syntax, and structure of metadata are formulated in the separate domains, so that metadata formats developed for one domain can be merged with formats developed for a second domain and used in a third domain without losing any of the clarity of the original statements. RDF is designed to make it easy for software to understand enough about a Web site so that it can discover resources on a site, catalog the site's content, rate that content, figure out who owns the content and under what terms and at what cost it may be used, and do other things a Web spider or intelligent agent might want to do.

RDF Statements

An RDF document or element makes statements about resources. A statement says that a certain resource has one or more properties. Each property has a type (that is, a name) and a value. The value of a property may be a literal such as a string, number, or date, or it may be another resource.

A statement can be thought of as a triple composed of three items: resource, property type, and property value. For example, an RDF statement might say, “The book *The XML Bible* (ISBN: 0-7645-3236-7) has the author Elliotte Rusty Harold.” Here the resource is “The book *The XML Bible* (ISBN: 0-7645-3236-7),” and the author property of this resource has the value “Elliotte Rusty Harold.” Figure 19-1 demonstrates a common way of pictorially describing this RDF statement.

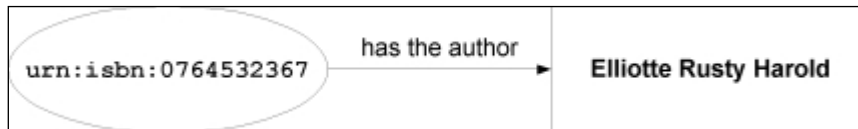


Figure 19-1: An RDF statement described in a picture

A resource can be anything that can have a Uniform Resource Identifier (URI). URIs are a superset of the more common Uniform Resource Locators (URLs), but they can also identify books, elements on a page, television shows, individual people, and more. In the above example, an ISBN is used as a URI for a book. Thus, a resource might be an entire Web site (<http://www.norm1.org/>), a single Web page (<http://www.mozilla.org/rdf/doc/index.html>), a specific HTML or XML element on a Web page identified with an XPointer ([http://metalab.unc.edu/xml/mail-inglists.html#root\(\).child\(1,dt\)](http://metalab.unc.edu/xml/mail-inglists.html#root().child(1,dt))), a book (`urn:isbn:0764532367`), a person (`mailto:elharo@metalab.unc.edu`), or just about anything — as long as a URI can be constructed for it. The only requirement for being a resource is a unique URI. This URI does not have to be a URL; it can be something else, such as an ISBN.

Resources are described with properties. A property is a specific characteristic, attribute, or relationship of a resource. Each property has a specific meaning that can be identified by the property's name and the associated schema. The schema should be found at the URI used for the property's namespace. The schema identifies the values, or value ranges, that are permitted for the property, and the types of resources it can describe.



Schemas are still in the development stages, so don't be too surprised if you don't actually find a schema where one is supposed to be. Also note that a namespace URI pointing to a schema is an RDF requirement, not a requirement of namespaces in general. In fact, the namespaces specification specifically denies any such requirement.

RDF only defines an XML syntax for encoding these resource-property type-property value triples in XML. It does not define the actual vocabularies used for describing resources and properties. Eventually this need will need to be addressed as well, at least if RDF is to be useful beyond a local intranet. Efforts are underway to produce standard vocabularies for content rating (PICS 2.0), personal information (P3P), and digital library catalogs (Dublin Core). Others can be invented as needed.

An RDF statement combines a specific resource with a named property and its value. These three parts of the statement are called, respectively, the subject, the predicate, and the object. The resource being described is the subject. The property used to describe the resource is the predicate. And the value of the property is the statement's object.

Here's a normal, human-readable statement:

*Elliotte Rusty Harold is the creator of the Web site at the URL
<http://metalab.unc.edu/xml/>.*

This same statement can be written in several other ways in English. For example:

The Web site at the URL <http://metalab.unc.edu/xml/> has the creator Elliotte Rusty Harold.

The Web site at the URL <http://metalab.unc.edu/xml/> was created by Elliotte Rusty Harold.

The creator of the Web site at the URL <http://metalab.unc.edu/xml/> is Elliotte Rusty Harold.

Elliotte Rusty Harold created the Web site at the URL <http://metalab.unc.edu/xml/>.

However, all five versions mean exactly the same thing. In each version, the subject is the Web site at the URL <http://metalab.unc.edu/xml/>. The predicate is the creator property. The object is the value of the creator property, Elliotte Rusty Harold. Figure 19-2 diagrams this statement as RDF understands it.

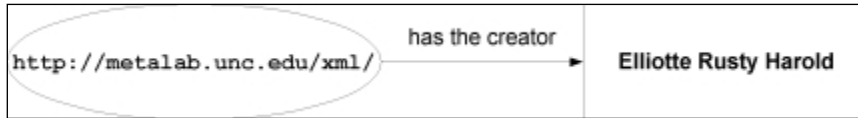


Figure 19-2: The statement in diagram form

Note

The RDF subject, object, and predicate do not correspond to the common use of those terms in English grammar. Indeed, part of the purpose of RDF is to separate the meaning of subject, object, predicate in an idea from their roles in any given sentence since the same idea can be expressed in multiple sentences, in each of which the grammatical subject, object, and predicate change places.

Basic RDF Syntax

The purpose of RDF is to take a meaningful statement such as “Elliote Rusty Harold is the creator of the Web site at the URL `http://metalab.unc.edu/xml/`” and write it in a standard XML format that computers can parse.

The root Element

The root element of an RDF document is `RDF`. This and all other RDF elements are normally placed in the `http://www.w3.org/1999/02/22-rdf-syntax-ns#` namespace. (As strange as it looks, the `#` is not a typo. It’s there so that when the element name is concatenated with the namespace, the result is a correct URL.) This namespace is either given the prefix `rdf` or set as the default namespace. For example, with an explicit prefix, an empty RDF element looks like this:

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!-- rdf:Description elements will go here -->
</rdf:RDF>
  
```

With the default namespace, it looks like this:

```

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!-- rdf:Description elements will go here -->
</RDF>
  
```

The Description Element

A RDF statement is serialized into XML encoded as a `Description` element. Each property of the resource being described is a child element of the `Description` element. The content of the child element is the value of the property. For example,

Listing 19-1 translates the statement “Elliotte Rusty Harold created the Web site at the URL `http://metalab.unc.edu/xml/`” into RDF.

Listing 19-1: The statement translated into RDF

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description about="http://metalab.unc.edu/xml/">
    <Creator>Elliotte Rusty Harold</Creator>
  </rdf:Description>
</rdf:RDF>
```

This `rdf:RDF` element contains a single statement. The statement is encoded as an `rdf:Description` element. The resource this statement is about (the subject) is `http://metalab.unc.edu/xml/`. The predicate of this statement is the content of the `rdf:Description` element, `<Creator>Elliotte Rusty Harold</Creator>`. The object of this statement is the content of the `Creator` element, `Elliotte Rusty Harold`. In short, the statement says that the resource at `http://metalab.unc.edu/xml/` has a `Creator` property whose value is the literal string `Elliotte Rusty Harold`.

Namespaces

Namespaces are used to distinguish between RDF elements and elements in property types and values. The `http://www.w3.org/1999/02/22-rdf-syntax-ns/` namespace is often used for RDF elements, generally with an `rdf` prefix. In the example above, the `Creator` element is in the default namespace. However, the descriptions may (and should) come from a different, nondefault namespace. For instance, the RDF element in Listing 19-2 uses the Dublin Core vocabulary and the `http://purl.org/DC/` namespace.

Listing 19-2: Elements from the Dublin Core vocabulary are in the namespace

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/DC/">
  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
  </rdf:Description>
</rdf:RDF>
```

The Dublin Core

The Dublin Core (<http://purl.org/dc/>) is a collection of elements designed to help researchers find electronic resources in a manner similar to using a library card catalog. Dublin Core elements include basic cataloging information, in particular:

- ♦ **TITLE:** The name given to the resource.
- ♦ **CREATOR:** The person or organization that created most of the resource (the author of a novel or the photographer who took a picture).
- ♦ **SUBJECT:** The topic of the resource.
- ♦ **DESCRIPTION:** A brief description of the resource, such as an abstract.
- ♦ **PUBLISHER:** The person or organization making the resource available (for example, IDG Books, Claremont University, or Apple Computer).
- ♦ **CONTRIBUTOR:** A non-CREATOR who contributed to the resource (the illustrator or editor of a novel).
- ♦ **DATE:** The date the resource was made available in its present form, generally in the format YYYY-MM-DD, such as 1999-12-31.
- ♦ **TYPE:** The category of the resource for example Web page, short story, poem, article, or photograph. Work is ongoing to produce a definitive list of acceptable resource types.
- ♦ **FORMAT:** The format of the resource, such as PDF, HTML, or JPEG. Work is ongoing to produce a definitive list of acceptable resource formats.
- ♦ **IDENTIFIER:** A unique string or number for the resource (as with a URL, a social security number, or an ISBN).
- ♦ **SOURCE:** A string or number that uniquely identifies the work from which the resource was derived. For instance, a Web page with the text of Jerome K. Jerome's 19th century novel *Three Men in a Boat* might use this to specify the specific edition from which text was scanned.
- ♦ **LANGUAGE:** The primary language in which the resource is written as ISO 639 language code.
- ♦ **RIGHTS:** Copyright and other intellectual property notices specifying the conditions under which the resource may or may not be used.

Several other possible Dublin Core elements are in the experimental stage including RELATION and COVERAGE. The Dublin Core is used throughout the examples in this chapter. However, you are by no means limited to using only these elements. You are free to use different vocabularies and namespaces for properties as long as you put them in a namespace.

Multiple Properties and Statements

A single `Description` element can state more than one property about a resource. For instance, what's missing from the previous statement is the name of the site, *Cafe con Leche*. A statement that includes this is, “Elliote Rusty Harold is the author of the *Cafe con Leche* Web site at the URL `http://metalab.unc.edu/xml/`.” Rewritten in more stilted, RDF-like syntax, this becomes “The Web site at the URL `http://metalab.unc.edu/xml/` has the name *Cafe con Leche* and was created by Elliote Rusty Harold.” Figure 19-3 diagrams this statement. Listing 19-3 shows how to add the property name to the RDF serialization in a natural way as simply one more child of `rdf:Description`, `dc:TITLE`.

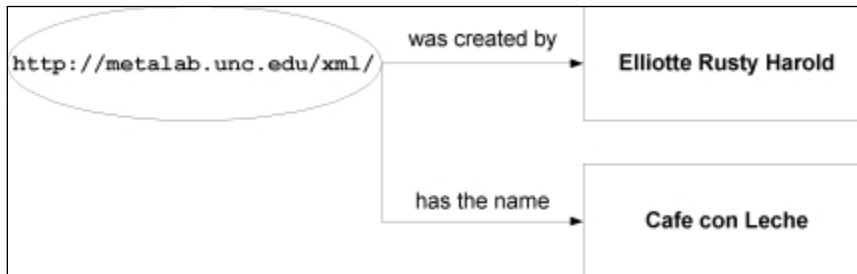


Figure 19-3: A statement with multiple properties in diagram form

Listing 19-3: A statement with multiple properties in RDF serialization form

```

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:CREATOR>Elliote Rusty Harold</dc:CREATOR>
    <dc:TITLE>Cafe con Leche</dc:TITLE>
  </rdf:Description>

</rdf:RDF>

```

A single RDF element can contain any number of `Description` elements, allowing it to make any number of statements. For example, suppose you want to make the two separate statements “Elliote Rusty Harold is the author of the *Cafe con Leche* Web site at the URL `http://metalab.unc.edu/xml/`” and “Elliote Rusty Harold is the author of the *Cafe au Lait* Web site at the URL `http://metalab.unc.edu/javafaq/`.” These are two statements about two different resources. Listing 19-4 shows how these are encoded in RDF.

Listing 19-4 Two separate statements encoded in RDF

```

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
    <dc:TITLE>Cafe con Leche</dc:TITLE>
  </rdf:Description>

  <rdf:Description about="http://metalab.unc.edu/javafaq/">
    <dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
    <dc:TITLE>Cafe au Lait</dc:TITLE>
  </rdf:Description>

</rdf:RDF>

```

Resource Valued Properties

A slightly more complicated example is the statement “The Cafe con Leche Web site at the URL <http://metalab.unc.edu/xml/> has the creator Elliotte Rusty Harold, whose email address is elharo@metalab.unc.edu.” The email address is the key. It provides a unique identifier for an individual, specifically the URL <mailto:elharo@metalab.unc.edu>. Thus, the individual becomes a resource rather than simply a literal. This resource is the value of the “created by” property of the <http://metalab.unc.edu/xml/> resource. Figure 19-4 diagrams this statement.

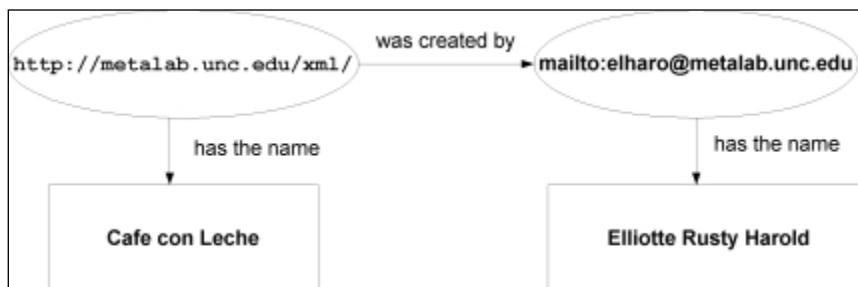


Figure 19-4: A statement with a resource valued property in diagram form

Encoding this statement in RDF is straightforward. Simply give the Creator element a Description child that describes the <mailto:elharo@metalab.unc.edu> resource, as in Listing 19-5.

Listing 19-5: A statement encoded in RDF with nested Description elements

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

  <Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR/>
    <Description about="mailto:elharo@metalab.unc.edu">
      <dc:TITLE>Elliotte Rusty Harold</dc:TITLE>
    </Description>
  </dc:CREATOR>
</Description>
</RDF>
```

There's no limit to the depth to which descriptions can be nested, nor is there any limit to the number of properties that can be applied to a Description element, nested or unnested.

RDF also provides an alternate syntax in which Description elements are not nested inside each other. Instead, the resource being described contains a resource attribute that points to the URI of the Description element. For example, Listing 19-6 is an equivalent serialization of the statement “The Cafe con Leche Web site at the URL <http://metalab.unc.edu/xml/> has the creator Elliotte Rusty Harold, whose email address is elharo@metalab.unc.edu.”

Listing 19-6: Descriptions by reference using the resource attribute

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR rdf:resource="mailto:elharo@metalab.unc.edu"/>
  </rdf:Description>

  <rdf:Description about="mailto:elharo@metalab.unc.edu">
    <dc:TITLE>Elliotte Rusty Harold</dc:TITLE>
  </rdf:Description>

</rdf:RDF>
```

Although this syntax is harder for a human reader to parse, it doesn't present any significant difficulties to a computer program. The primary advantage is that it allows the same property to be attached to multiple resources. For example, consider the statement "Elliote Rusty Harold, whose email address is `elharo@metalab.unc.edu`, created both the Cafe con Leche Web site at the URL `http://metalab.unc.edu/xml/` and the Cafe au Lait Web site at the URL `http://metalab.unc.edu/javafaq/`," which is diagrammed in Figure 19-5. This is easily serialized, as shown in Listing 19-7. The description of the resource `mailto:elharo@metalab.unc.edu` does not have to be repeated.

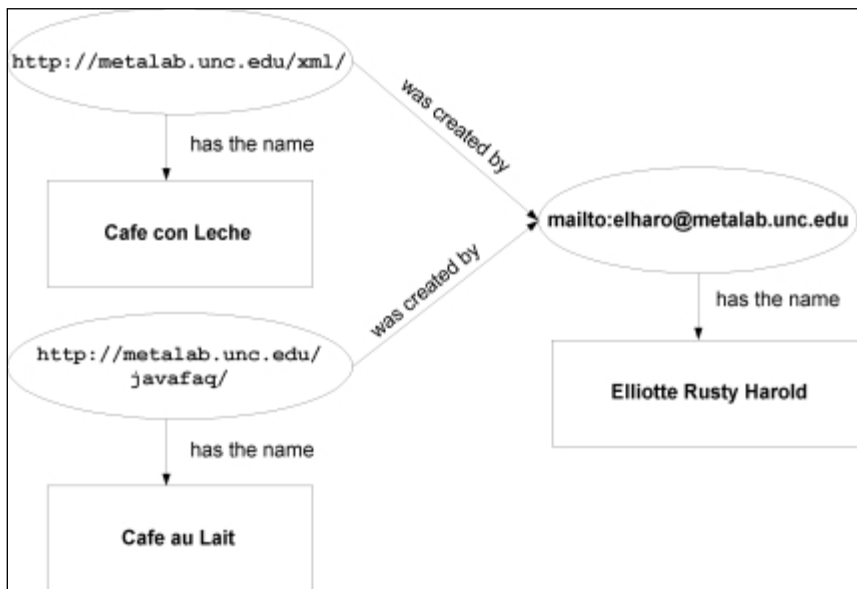


Figure 19-5: The statement in diagram form

Listing 19-7: A statement with the same property attached to multiple resources

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
```

```

    <dc:CREATOR rdf:resource="mailto:elharo@metalab.unc.edu"/>
  </rdf:Description>

  <rdf:Description about="http://metalab.unc.edu/javafaq/">
    <dc:TITLE>Cafe au Lait</dc:TITLE>
    <dc:CREATOR rdf:resource="mailto:elharo@metalab.unc.edu"/>
  </rdf:Description>

  <rdf:Description about="mailto:elharo@metalab.unc.edu">
    <dc:TITLE>Elliotte Rusty Harold</dc:TITLE>
  </rdf:Description>

</rdf:RDF>

```

XML Valued Properties

Property values are most commonly either pure text or resources. However, they may also contain well-formed XML markup that is not itself RDF markup. In this case, the property element must have a `parseType` attribute with the value `Literal`, as shown in Listing 19-8.

Listing 19-8: A literal property value that uses XML markup

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://www.purl.org/DC/"
  xmlns:nm="http://www.metalab.unc.edu/xml/names/">

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:CREATOR parseType="Literal">
      <nm:FirstName>Elliotte</nm:FirstName>
      <nm:MiddleName>Rusty</nm:MiddleName>
      <nm:LastName>Harold</nm:LastName>
    </dc:CREATOR>
  </rdf:Description>

</rdf:RDF>

```

Without `parseType="Literal"`, the value of a property must be a resource or parsed character data only. It must not contain any embedded markup.

Abbreviated RDF Syntax

As well as the basic syntax used above, RDF also defines an abbreviated syntax that uses attributes instead of parsed character data content. This is convenient when RDF data is embedded in an HTML page, because a Web browser can simply ignore the RDF tags without any affect on the rendered page. The two syntaxes are completely equivalent from the perspective of an RDF (as opposed to HTML) parser.

In abbreviated syntax, each property becomes an attribute of the `Description` element. The name of the property is the name of the attribute. If the property has a literal value, the value of the property is the value of the attribute. If the property has a resource value, the value of the property is the URI of the resource, and a separate `Description` element describes the resource. Because the `Description` element no longer has a variety of child elements, it does not need a closing tag and is written using normal empty element syntax.

The simple statement “Elliote Rusty Harold created the Web site `http://metalab.unc.edu/xml/`” is written in abbreviated form, like this:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/DC/">
  <Description about="http://metalab.unc.edu/xml/"
    dc:CREATOR="Elliote Rusty Harold" />
</RDF>
```

The statement “Elliote Rusty Harold created the Cafe con Leche Web site `http://metalab.unc.edu/xml/`” is written in abbreviated form, like this:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/DC/">
  <Description about="http://metalab.unc.edu/xml/"
    dc:CREATOR="Elliote Rusty Harold"
    dc:TITLE="Cafe con Leche" />
</RDF>
```

Resource valued properties are trickier to abbreviate. The statement “The Cafe con Leche Web site at the URL `http://metalab.unc.edu/xml/` has the creator Elliote Rusty Harold, whose email address is `elharo@metalab.unc.edu`” can be abbreviated like this:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/DC/">
  <rdf:Description about="http://metalab.unc.edu/xml/"
```

```
dc:TITLE="Cafe con Leche">
  <dc:CREATOR rdf:resource="mailto:elharo@metalab.unc.edu"
    dc:TITLE="Elliotte Rusty Harold" />
</rdf:Description>
</rdf:RDF>
```

Here the `Description` element is nonempty because it has a `Creator` child. However, it still doesn't contain any character data except white space.

Containers

When an RDF element describes a resource with multiple properties of the same type, for example to say that a document was written by multiple people or to list mirror sites where a Web page can be found, a container can group the property values. Every item in the group is a property value of the same type (property name). This allows you to describe the group as a whole rather than merely describe individual items in the container. RDF defines three types of container objects:

1. `Bag`: a group of unordered properties
2. `Seq`: a sequence (ordered list) of properties
3. `Alt`: a list of alternative properties from which a single one is chosen

The Bag container

A bag is a list of property values (resources and literals), in no particular order, all of which share the same property name (type). This allows you to declare a property that has more than one value, such as the authors of a book or the members of a committee. A bag may contain duplicate values.

A bag of properties is represented by a `Bag` element. Each item in the bag is a `li` child element of the `Bag`. The `Bag` itself is a child of the `Description` to which it applies.

For example, consider the statement “The Cafe con Leche Web site at <http://metalab.unc.edu/xml/> was created by Elliotte Rusty Harold to provide XML news, XML mailing lists, XML conferences, and XML books.” This is diagrammed in Figure 19-6. The four main subjects of the site can be collected in a `Bag`, as shown in Listing 19-9.

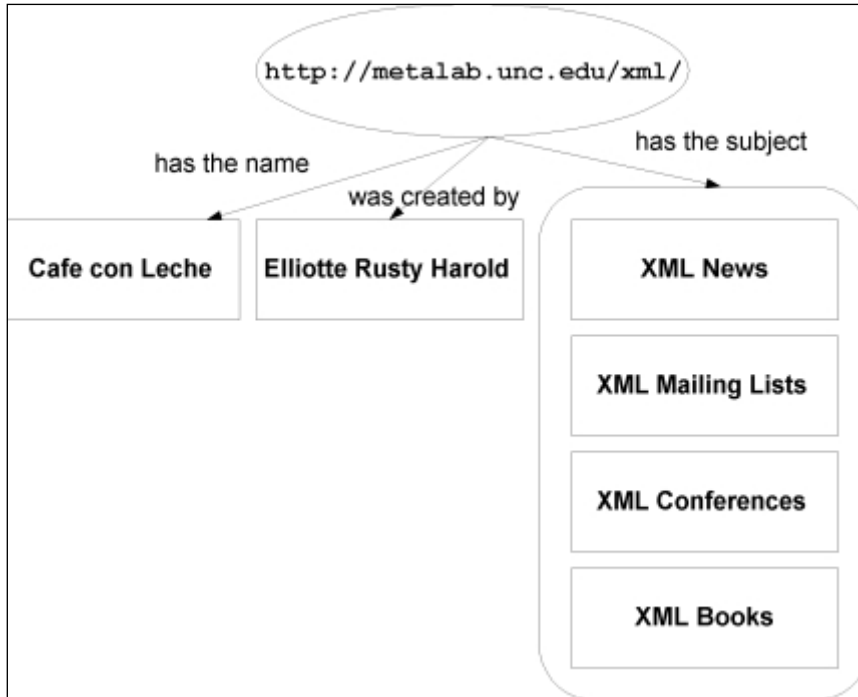


Figure 19-6: The statement in diagram form

Listing 19-9: A bag with four members

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR>Elliote Rusty Harold</dc:CREATOR>
    <dc:SUBJECT>
      <rdf:Bag>
        <rdf:li>XML News</rdf:li>
        <rdf:li>XML Mailing lists</rdf:li>
        <rdf:li>XML Conferences</rdf:li>
        <rdf:li>XML Books</rdf:li>
      </rdf:Bag>
    </dc:SUBJECT>
  </rdf:Description>

</rdf:RDF>
```

If the members of the bag are resources rather than literals, they're identified with a resource attribute. For example, Listing 19-10 provides a simple site map for Cafe con Leche.

Listing 19-10: A simple site map for Cafe con Leche in a Bag

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR>Elliott Rusty Harold</dc:CREATOR>
    <dc:SUBJECT>
      <rdf:Bag>
        <rdf:li
          resource="http://metalab.unc.edu/xml/news1999.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/maillinglists.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/news1999.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/tradeshows.html"/>
      </rdf:Bag>
    </dc:SUBJECT>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/news1999.html">
    <dc:TITLE>XML News from 1999</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/books.html">
    <dc:TITLE>XML Books</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/maillinglists.html">
    <dc:TITLE>XML Mailing Lists</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/tradeshows.html">
    <dc:TITLE>XML Trade Shows and Conferences</dc:TITLE>
  </rdf:Description>

</rdf:RDF>
```

The Seq Container

A sequence container is similar to a bag container. However, it guarantees that the order of the contents is maintained. Sequences are written exactly like bags, except that the `Seq` element replaces the `Bag` element. For example, this sequence guarantees that when the `Subject` is read out by an RDF parser, it comes out in the order “XML News, XML Mailing Lists, XML Conferences, XML Books” and not some other order such as “XML Books, XML Conferences, XML Mailing Lists, XML News.”

```
<dc:SUBJECT>
  <rdf:Seq>
    <rdf:li>XML News</rdf:li>
    <rdf:li>XML Mailing lists</rdf:li>
    <rdf:li>XML Conferences</rdf:li>
    <rdf:li>XML Books</rdf:li>
  </rdf:Seq>
</dc:SUBJECT>
```

In practice, maintaining the order of properties in a container is rarely important, so sequences aren't used as much as bags and alternatives.

The Alt Container

The `Alt` container includes one or more members from which a single one is picked. For example, this might be used to describe the mirrors of a Web site. Consider the statement “The Cafe au Lait Web site at <http://metalab.unc.edu/javafaq/> created by Elliotte Rusty Harold is mirrored at [Sunsite Austria](http://sunsite.univie.ac.at/jcca/mirrors/javafaq/) (<http://sunsite.univie.ac.at/jcca/mirrors/javafaq/>), [Sunsite Slovakia](http://sunsite.uakom.sk/javafaq/) (<http://sunsite.uakom.sk/javafaq/>), [Sunsite Sweden](http://sunsite.kth.se/javafaq/) (<http://sunsite.kth.se/javafaq/>), and [Sunsite Switzerland](http://sunsite.cnlab-switch.ch/javafaq/) (<http://sunsite.cnlab-switch.ch/javafaq/>).” Because only one of these mirror sites is desired, they can be placed in an alternative list. Listing 19-11 shows the RDF serialization.

Listing 19-11: Mirror sites of Cafe au Lait in a Seq

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
    <dc:PUBLISHER>
      <rdf:Alt>
        <rdf:li resource =
```

```

        "http://sunsite.univie.ac.at/jcca/mirrors/javafaq/" />
<rdf:li resource =
  "http://sunsite.kth.se/javafaq/" />
<rdf:li resource =
  "http://sunsite.cnlab-switch.ch/javafaq/" />
<rdf:li resource =
  "http://sunsite.uakom.sk/javafaq/" />
</rdf:Alt>
</dc:PUBLISHER>
</rdf:Description>

<rdf:Description
  about="http://sunsite.univie.ac.at/jcca/mirrors/javafaq/">
  <dc:PUBLISHER>Sunsite Austria</dc:PUBLISHER>
</rdf:Description>

<rdf:Description
  about="http://sunsite.uakom.sk/javafaq/">
  <dc:PUBLISHER>Sunsite Slovakia</dc:PUBLISHER>
</rdf:Description>

<rdf:Description
  about="http://sunsite.cnlab-switch.ch/javafaq/">
  <dc:PUBLISHER>Sunsite Switzerland</dc:PUBLISHER>
</rdf:Description>

<rdf:Description
  about="http://sunsite.kth.se/javafaq/">
  <dc:PUBLISHER>Sunsite Sweden</dc:PUBLISHER>
</rdf:Description>

</rdf:RDF>

```

Statements about Containers

Statements can be made about a container as a whole, separate from statements about individual items in the container. You may want to say that a particular person developed a Web site without implying that he or she personally wrote each and every page on the site. Or perhaps you want to claim a copyright on a collection of links without claiming a copyright on the pages to which you're linking. (For example, the market values Yahoo's collection of links and descriptions at several hundred million dollars, even though Yahoo owns essentially none of the pages to which it links.) In fact, the individual members of the container might have different copyrights than the container itself. Figure 19-7 diagrams this.

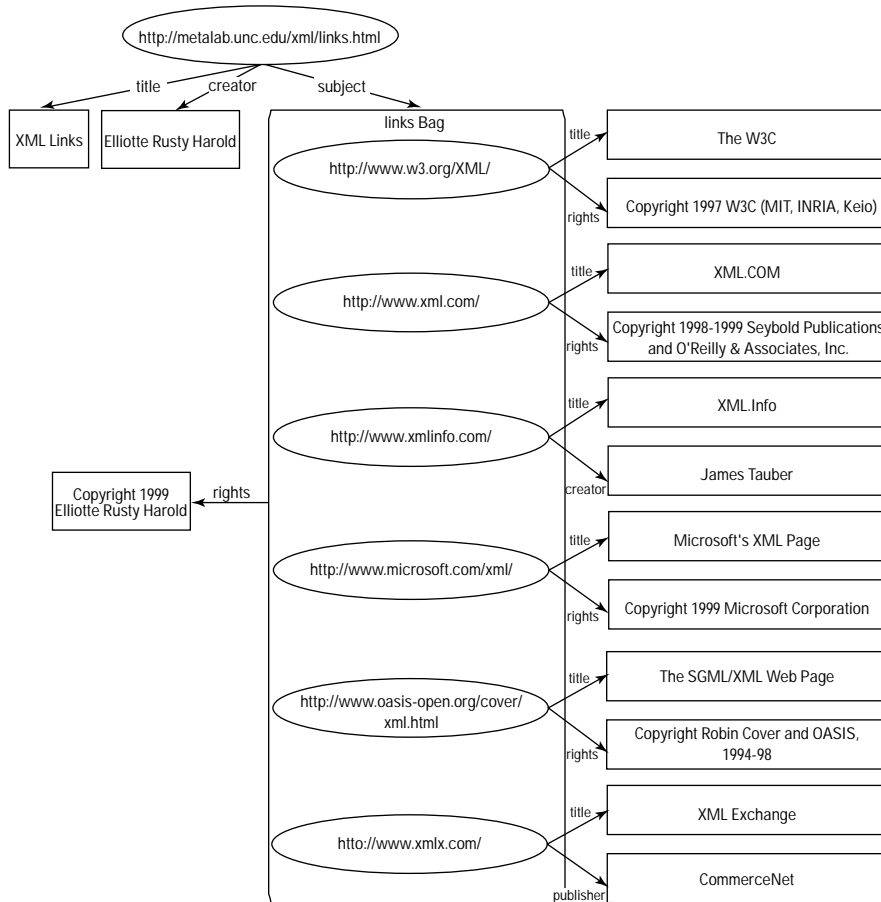


Figure 19-7: A bag whose rights information is different than the rights information of the individual members of the bag

To encode this in RDF, give the container (Bag, Seq, or Alt) an ID attribute. Description elements with about attributes, whose value is a relative URL pointing to the container ID, describe the container.

Listing 19-12: A description of a container encoded in RDF

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

  <rdf:Description
    about="http://metalab.unc.edu/xml/links.html">
```

```

<dc:TITLE>XML Links</dc:TITLE>
<dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
<dc:SUBJECT>
  <rdf:Bag ID="links">
    <rdf:li resource="http://www.w3.org/XML/" />
    <rdf:li resource="http://www.xml.com/" />
    <rdf:li resource="http://www.xmlinfo.com/" />
    <rdf:li resource="http://www.microsoft.com/xml/" />
    <rdf:li
      resource="http://www.oasis-open.org/cover/xml.html" />
    <rdf:li resource="http://www.xmlx.com/" />
  </rdf:Bag>
</dc:SUBJECT>
</rdf:Description>

<rdf:Description about="#links">
  <dc:RIGHTS>
    Copyright 1999 Elliotte Rusty Harold
  </dc:RIGHTS>
</rdf:Description>

<rdf:Description about="http://www.w3.org/XML/">
  <dc:TITLE>The W3C</dc:TITLE>
  <dc:RIGHTS>
    Copyright 1997 W3C (MIT, INRIA, Keio)
  </dc:RIGHTS>
</rdf:Description>

<rdf:Description about="http://www.xml.com/">
  <dc:TITLE>xml.com</dc:TITLE>
  <dc:RIGHTS>
    Copyright 1998-1999 Seybold Publications
    and O'Reilly & Associates, Inc.
  </dc:RIGHTS>
</rdf:Description>

<rdf:Description about="http://www.xmlinfo.com/">
  <dc:TITLE>XML Info</dc:TITLE>
  <dc:CREATOR>James Tauber</dc:CREATOR>
</rdf:Description>

<rdf:Description about="http://www.microsoft.com/xml/">
  <dc:TITLE>Microsoft's XML Page</dc:TITLE>
  <dc:RIGHTS>Copyright 1999 Microsoft Corporation</dc:RIGHTS>
</rdf:Description>

<rdf:Description
  about="http://www.oasis-open.org/cover/xml.html">
  <dc:TITLE>Robin Cover's XML Web Page</dc:TITLE>
  <dc:RIGHTS>
    Copyright Robin Cover and OASIS, 1994-98

```

Continued

Listing 19-12 (continued)

```

    </dc:RIGHTS>
  </rdf:Description>

  <rdf:Description about="http://www.xmlx.com/">
    <dc:TITLE>XML Exchange</dc:TITLE>
    <dc:PUBLISHER>CommerceNet</dc:PUBLISHER>
  </rdf:Description>

</rdf:RDF>

```

Statements about Container Members

Sometimes you do want to make a statement about each member of a container, but you don't want to repeat the same description three or four times. For example, you may want to specify that the title and creator of each of the mirror sites is Cafe au Lait and Elliott Rusty Harold, respectively, as shown in Figure 19-8.

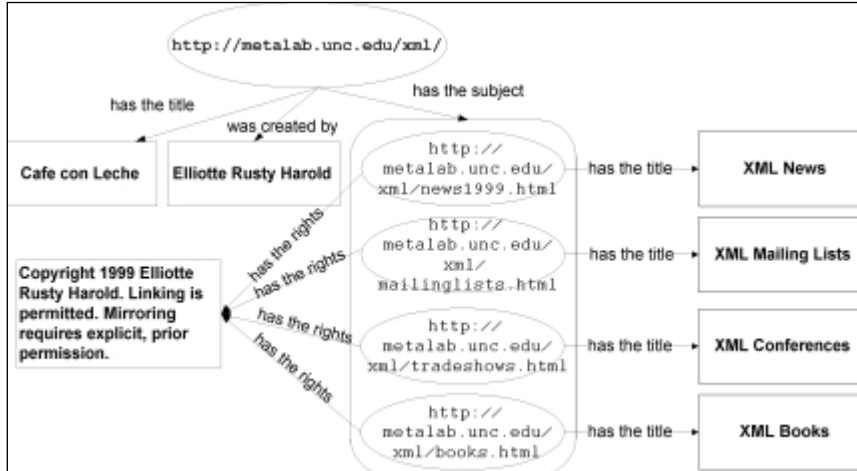


Figure 19-8: Attaching the same description to each page in the bag

You can include an `aboutEach` attribute in the `Bag`, `Seq`, or `Alt` element whose value is a name by which descriptions can be applied to all the members of the container. For example, suppose you want to apply a copyright notice to each page in a `Bag`. Listing 19-13 accomplishes this.

Listing 19-13: A description of each element in a Bag container

```

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

  <rdf:Description about="http://metalab.unc.edu/xml/">
    <dc:TITLE>Cafe con Leche</dc:TITLE>
    <dc:CREATOR>Elliotte Rusty Harold</dc:CREATOR>
    <dc:SUBJECT>
      <rdf:Bag aboutEach="pages">
        <rdf:li
          resource="http://metalab.unc.edu/xml/news1999.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/maillinglists.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/news1999.html"/>
        <rdf:li
          resource="http://metalab.unc.edu/xml/tradeshows.html"/>
      </rdf:Bag>
    </dc:SUBJECT>
  </rdf:Description>

  <rdf:Description aboutEach="#pages">
    <dc:RIGHTS>
      Copyright 1999 Elliotte Rusty Harold
      Linking is permitted.
      Mirroring requires explicit, prior permission.
    </dc:RIGHTS>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/news1999.html">
    <dc:TITLE>XML News from 1999</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/books.html">
    <dc:TITLE>XML Books</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/maillinglists.html">
    <dc:TITLE>XML Mailing Lists</dc:TITLE>
  </rdf:Description>

  <rdf:Description
    about="http://metalab.unc.edu/xml/tradeshows.html">
    <dc:TITLE>XML Trade Shows and Conferences</dc:TITLE>
  </rdf:Description>

</rdf:RDF>

```

Statements about Implied Bags

Sometimes you want to make a statement about a group of resources that may or may not be members of the same container. For example, suppose you want to specify that every page on the Web site `http://www.macfaq.com` is “Copyright 1999 Elliotte Rusty Harold.” You can do this with a `Description` element that applies to all resources whose URI begins with the string “`http://www.macfaq.com`”. This `Description` element must have an `aboutEachPrefix` attribute whose value is the URI prefix of the resources to which the description applies. For example:

```
<rdf:Description aboutEachPrefix="#http://www.macfaq.com">
  <dc:RIGHTS>Copyright 1999 Elliotte Rusty Harold</dc:RIGHTS>
</rdf:Description>
```

This `Description` element creates an implicit bag whose members are the resources matching the prefix. These resources may or may not be members of other containers in the RDF file, and they may or may not be sibling elements. The members of this implied bag are gathered from wherever they reside.

URI prefixes can be used to select only a subtree of a Web site. For example, this description claims that all pages at `metalab.unc.edu` in the `/xml` hierarchy are “Copyright 1999 Elliotte Rusty Harold”. However, it does not apply to other pages outside that hierarchy such as `http://metalab.unc.edu/id/asylum` or `http://metalab.unc.edu/stats/`.

```
<rdf:Description
  aboutEachPrefix="#http://metalab.unc.edu/xml/">
  <dc:RIGHTS>Copyright 1999 Elliotte Rusty Harold</dc:RIGHTS>
</rdf:Description>
```

For another example, take ISBNs assigned by publishers. All books from IDG Books have an ISBN that begins 07645. Thus, this `Description` element creates an implicit Bag containing only books published by IDG Books and assigns a `Publisher` property to each member:

```
<rdf:Description aboutEachPrefix="#urn:isbn:07645">
  <dc:PUBLISHER>IDG Books</dc:PUBLISHER>
</rdf:Description>
```

RDF Schemas

Although there’s no guarantee that a generic XML namespace URI points to anything in particular, RDF is stricter than that. Any namespace URI used in RDF should point to a schema for the vocabulary. The schema describes the semantics and allowed syntax of a particular element. For instance, the schema may say that the contents

of a DATE element must be in the form 1999-12-31 and not in the form December 31, 1999. A schema may also make DTD-like statements, such as that each BOOK element must contain one or more AUTHOR child elements.

Exactly how a schema makes statements such as this is a subject of debate. In practice, current RDF schemas are mostly written in prose that human beings read. For example, part of the Dublin Core “schema” is shown in Figure 19-10. (In the long run, a more formal and complete schema for the Dublin Core is likely to be developed.)

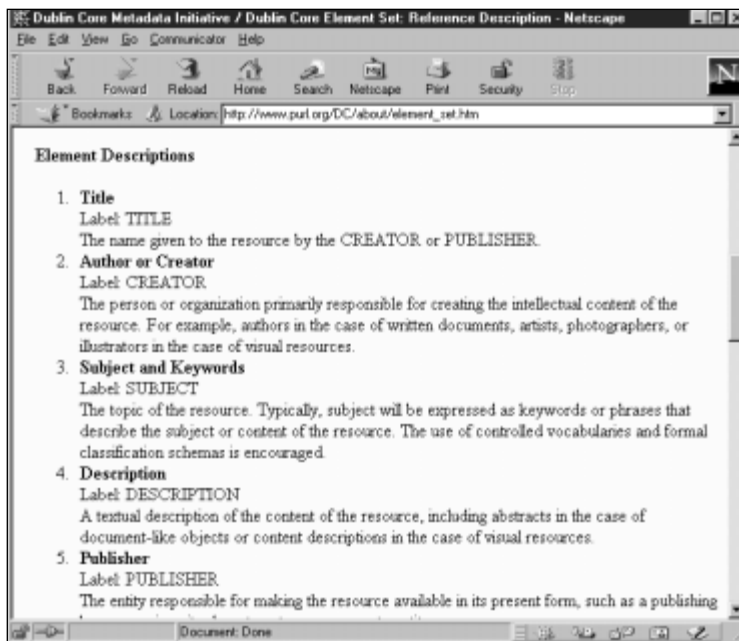


Figure 19-9: The Dublin Core schema

Eventually schemas will be written in a more formal syntax that computers can understand. In particular, the W3C RDF Schema Working Group is attempting to develop an RDF schema specification that writes RDF schema in RDF. This will enable an RDF processor to validate a particular RDF document against the schemas it uses. However, this work is far from finished as of spring, 1999. If you're curious about this work, you can retrieve the current draft of the RDF schema specification from <http://www.w3.org/TR/1998/WD-rdf-schema/>.

Summary

This chapter covered RDF. In particular, you learned:

- ♦ The Resource Description Framework (RDF) is an XML application for structured metadata. Metadata is information about information.
- ♦ An RDF document or element makes statements about resources.
- ♦ Each statement specifies a resource, a property of that resource, and the value of that property.
- ♦ A resource is anything that has a Uniform Resource Identifier (URI). URLs of Web pages are just one form of URI.
- ♦ The value of a property may be plain text, another resource, or XML markup.
- ♦ The root element of an RDF document is `RDF`.
- ♦ An RDF element contains `Description` elements that make statements about resources.
- ♦ Each `Description` element contains either a literal property or a resource attribute whose value is the URI of the property value.
- ♦ RDF also defines an abbreviated syntax in which properties may be replaced by attributes of the same name on the `Description` element.
- ♦ The `Bag`, `Seq`, and `Alt` elements provide containers for multiple resources. Properties can be applied to the container as a whole, to the individual elements of the container, or both.
- ♦ The namespace URI for each vocabulary used in an RDF document should point to a schema for the vocabulary.

The next chapter starts to explain a number of other XML applications. It begins with an in-depth analysis of the Voyager HTML-in-XML DTD to help develop your skills at reading DTDs written by others.

