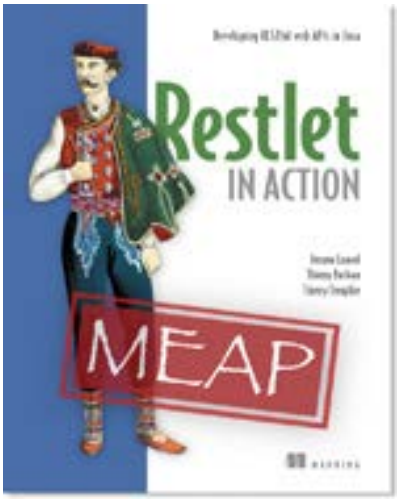
MANNING PUBLICATIONS



Restlet in Action
By Jerome Louvel, Thierry Templier, and Thierry Boileau

Hypermedia is not only about hypertext media types but also about hyperdata and, more importantly, about interoperability of data. This goal is shared by the Semantic Web, which was initiated by Tim Berners Lee. In this article, based on chapter 10 of Restlet in Action, authors introduce the Semantic Web and its relationship to REST via the new Linked Data trend, which is a great illustration of hyperdata. Then, they will introduce RDF, the core standard for semantic representations, and explain how to expose, consume, and browse linked data with Restlet.

Leveraging the Semantic Web with Linked Data

Hypermedia is not only about hypertext media types but also about hyperdata and, more importantly, about interoperability of data. This goal is shared by the Semantic Web, which was initiated by Tim Berners Lee.

In this article, we will first give a brief overview of the Semantic Web and its relationship to REST via the new Linked Data trend, which is a great illustration of hyperdata. Then, we will introduce RDF, the core standard for semantic representations, and explain how to expose, consume, and browse linked data with Restlet. This will also be the time to introduce FOAF, a RDF vocabulary to describe social relationships.

REST and the Semantic Web

The Semantic Web is an ambitious initiative that was publicly launched in 2001 by a now famous Scientific American article ("The Semantic Web" by Tim Berners-Lee, James Hendler and Ora Lassila) and that resulted in great expectations. However, the work on its foundations took a significant time and involved many researchers across the globe. During this time, developers and companies were left wondering how they could best leverage all those impressive specifications that started to come out of the W3C such as RDF, RDF Schema, OWL, or SPARQL. An impression of excessive complexity started to emerge along with the perceived lack of real-world use cases where the Semantic Web could shine.

That's when Berners-Lee introduced his Linked Data idea, as an application of the Semantic Web that would allow browsing semantically linked resources. Instead of storing semantic data in large specialized databases and requiring to use a special SPARQL language to interact with them as in the relational world, the idea was simply to use the Web and its HTTP protocol as a way to directly interact with the graph of semantic data, using hyperlinks to jump from one hyperdata document to another—just like we do with hypertext documents.

Linked Data finally offers a pragmatic and operational approach to the Semantic Web that is in addition perfectly in line with REST principles and especially the hypermedia one. This lighter approach was the foundation of Restlet support for the Semantic Web that is available in the `org.restlet.ext.rdf` extension.

Restlet and its semantic roots

When the Restlet Framework launched in 2005, it was the result of extracting a generic piece of code from a web site project called "Semalink" that aimed at facilitating the adoption of the Semantic Web by closing the gap with the regular Web of documents where REST played a pivotal role. Later on, the success of the Linked Data initiative and its support in Restlet via the RDF extension was in a way a return to the project roots.

Concretely, Linked Data relies on URIs (actually HTTP URLs) to identify important data, like we use them to identify important documents on the regular Web. It also relies on HTTP to retrieve, create, update or delete those data like for other RESTful web APIs.

As illustrated in figure 1, on top of its REST, URI, and HTTP lower layers, Linked Data relies on the RDF language to represent those data resources and their relationships with other resources or the literal values of their attributes. Finally, we have RDF Schema and its richer Ontology Web Language (OWL) cousin, which are languages (also expressed in RDF) used to define valid RDF graph structures called ontologies, or metamodels if you are more familiar with model-driven engineering.

In the long term, we can expect other layers of the Semantic Web vision to find their way inside this stack to solve issues such as distributed queries, proof and trust. There are already propositions to address those needs, such as SPARQL to query RDF databases, but some were defined before the Linked Data trend and will likely have to be rethought to better fit with the regular Web in order to reach a broader usage level.

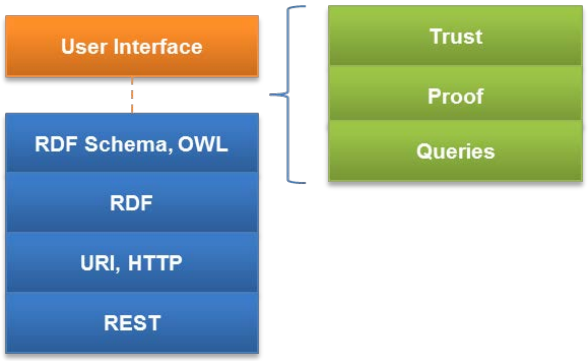


Figure 1 The Linked Data technological stack

Regarding trust, there is a WebID protocol in the W3C Incubator that aims at leveraging HTTPS and especially client SSL certificates to build a web of trust, in a pragmatic way that can nicely complements Linked Data. It is now time to have a closer look at it and see how it can be used as for resource representations.

Using RDF in representations

RDF is the acronym for Resource Description Framework, where a resource has the same meaning as in the REST architecture style, which is something of interest that can be addressed by a URI. As its names implies, RDF provides a way to describe and represent web resources in a semantic way—in a precise and interoperable way. To understand how RDF works, we will first present the RDF data model, explaining the main concepts involved and how they relate to REST, then the various serialization media types available for this data model.

To make this discussion more concrete, we will use examples from the FOAF language, which lets you express social links between peoples, a much simpler but open and semantic variant of Facebook or LinkedIn data sets.

RDF data model

In RDF, all of the data is defined as a graph, where nodes are either resources identified by an URI or literals (like a string or an Integer) and where links connect either one resource to another or a resource and a literal defined as an attribute value. Those links are also frequently named statements, triples, or properties.

In figure 2, we partially describe a sample (Homer Simpson) resource as graph with a central resource node, three literal nodes on the left, and three related resource nodes on the right.

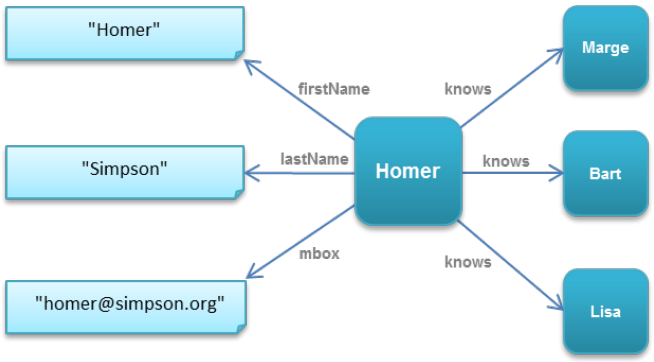


Figure 2 A sample RDF graph partially describing Homer Simpson

As you can see, the links have labels defining how Homer relates to the Marge, Bart, and Lisa resources or the meaning of the "`Homer`", "`Simpson`" and "`homer@simpson.org`" literals. Obviously, Homer knows his wife and children, and we could have added more links to express the exact familial relationships such as father and husband. However, in this case, we have decided to follow the FOAF vocabulary, which isn't especially interested in family relationships but mostly in generic links between people.

With the above example, we almost have a valid RDF graph. The piece of information missing is unambiguous information telling us that the links are really related to FOAF and not to another vocabulary. For this purpose, RDF relies again on URIs to precisely and uniquely define the meaning of those links.

This is the most important difference compared to hyperlinks found in HTML documents for example and provides a better interoperability and the ability to mix several links between the same two resources. In our case, the exact value of "knows" is http://xmlns.com/foaf/0.1/knows, the value of "mbox" is http://xmlns.com/foaf/0.1/mbox and so on.

Let's now take this example RDF graph and create it using the RDF extension of the Restlet Framework available in the `org.restlet.ext.rdf.jar` file. As illustrated in listing 1 below, the translation is very straightforward, especially as we are able to use the `Reference` class from the `org.restlet.data` package that defines URI references.

Listing 1 Creating a RDF graph with Restlet RDF extension

```
import org.restlet.data.Reference;
import org.restlet.ext.rdf.Graph;
import org.restlet.ext.rdf.Literal;

public class FoafExample {

    public static void main(String[] args) throws IOException {
        String FOAF_BASE = "http://xmlns.com/foaf/0.1/";    #A
        Reference firstName = new Reference(FOAF_BASE + "firstName");
        Reference lastName = new Reference(FOAF_BASE + "lastName");
        Reference mbox = new Reference(FOAF_BASE + "mbox");

        Reference homerRef = new Reference(                    #B
        "http://www.rmep.org/accounts/chunkylover53/");
        Reference margeRef = new Reference(
        "http://www.rmep.org/accounts/bretzels34/");
        Reference bartRef = new Reference(
        "http://www.rmep.org/accounts/jojo10/");
        Reference lisaRef = new Reference(
        "http://www.rmep.org/accounts/lisa1984/");
```

```
        Graph example = new Graph();        #C
        example.add(homerRef, firstName, new Literal("Homer"));
        example.add(homerRef, lastName, new Literal("Simpson"));
        example.add(homerRef, mbox, new Literal("mailto:homer@simpson.org"));
        example.add(homerRef, knows, margeRef);
        example.add(homerRef, knows, bartRef);
        example.add(homerRef, knows, lisaRef);
    }
```

#A FOAF ontology constants
#B Linked Simpson resources
#C Example RDF graph

As introduced in figure 1, it is also possible to define the structure of valid RDF graphs as ontologies using the RDF Schema and OWL languages. Ontology might sound like an impressive term, but in the end it is really just like a set of object classes where relations between classes as well as attributes are typed using not just a label but a precise and unambiguous URI.

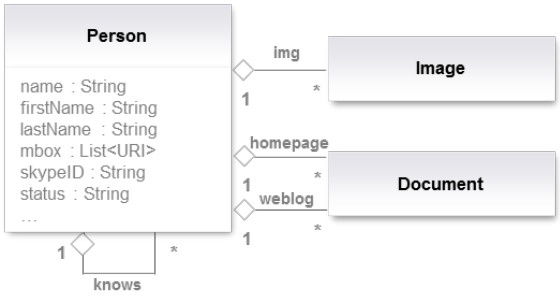


Figure 3 Person class in the FOAF vocabulary

In figure 3, we illustrated how to visualize a part of the FOAF ontology as a regular UML class diagram. An RDF class defined in a Linked Data way can be exactly the same as a REST class defined in a web API, only adding further information about its attributes and relationships with other resource classes, such as other persons, images, and documents in the FOAF case.

Let's now move beyond the abstract RDF data model and see how to serialize RDF graphs and use them as representations.

RDF representation variants

Contrary to XML vocabularies such as Atom or XHTML, RDF doesn't force you to use a single serialization format. Actually, even though the primary format is XML based, there are other ones available:

- RDF/XML, a comprehensive XML serialization format for RDF.
- Notation 3 (or just n3), a compact alternative to RDF/XML also able to express rules.
- Turtle, a subset of n3, simple and human readable.
- N-Triples, an even simpler subset of Turtle, useful for storing and exchanging RDF.

To make those formats more concrete, we will now serialize the example FOAF graph built with the Restlet extension. For this purpose, let's add the lines of code below to our code in listing 1.

```
System.out.println("\nRDF/XML format:\n");
example.getRdfXmlRepresentation().write(System.out);

System.out.println("\nRDF/n3 format:\n");
example.getRdfN3Representation().write(System.out);

System.out.println("\nRDF/Turtle format:\n");
example.getRdfTurtleRepresentation().write(System.out);

System.out.println("\nRDF/NTriples format:\n");
```

```
example.getRdfNTriplesRepresentation().write(System.out);
```

The `getRdf*Representation()` methods simply create an instance of the `RdfRepresentation` class passing it the `Graph` instance and the proper media type constant. Now, if we run this code, we will first serialize the graph into RDF/XML. The key XML element is "`rdf:Description`", which contains all the properties related to the Homer resource identified by the XML attribute "`xml:about`". Note also how a prefix "`__NS1`" was declared for the FOAF ontology URI.

```
<?xml version="1.0" standalone='yes'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:__NS1="http://xmlns.com/foaf/0.1/">
   <rdf:Description rdf:about="http://www.rmep.org/accounts/chunkylover53/">
      <__NS1:firstName>Homer</__NS1:firstName>
      <__NS1:lastName>Simpson</__NS1:lastName>
      <__NS1:mbox>mailto:homer@simpson.org</__NS1:mbox>
      <__NS1:knows>
         <rdf:Description rdf:about="http://www.rmep.org/accounts/bretzels34/"/></__NS1:knows>
      <__NS1:knows>
         <rdf:Description rdf:about="http://www.rmep.org/accounts/jojo10/"/></__NS1:knows>
      <__NS1:knows>
         <rdf:Description rdf:about="http://www.rmep.org/accounts/lisa1984/"/></__NS1:knows>
   </rdf:Description>
</rdf:RDF>
```

The second and third serializations are for n3 and Turtle formats, and produce the same result below. Note that the all the graph is written in just one line starting with `<http://www.rmep.org/accounts/chunkylover53/>`. In addition, namespace prefixes can be used as illustrated with the first lines below.

```
@prefix #: <:>.
@prefix rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix type: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf: <http://www.w3.org/2000/01/rdf-schema#>.
@keywords a, is, of, has.
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/firstName> "Homer";
<http://xmlns.com/foaf/0.1/lastName> "Simpson"; <http://xmlns.com/foaf/0.1/mbox>
"mailto:homer@simpson.org"; <http://xmlns.com/foaf/0.1/knows>
<http://www.rmep.org/accounts/bretzels34/>, <http://www.rmep.org/accounts/jojo10/>,
<http://www.rmep.org/accounts/lisa1984/>.
```

Finally, let's see how the fourth and simplest format, N-Triples, serializes our graph. For each link, there is a new line, and no prefixes or factorizations are used. The result is more verbose, but also very straightforward to write, read, and understand.

```
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/firstName> "Homer".
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/lastName> "Simpson".
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/mbox>
"mailto:homer@simpson.org".
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/knows>
<http://www.rmep.org/accounts/bretzels34/>.
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/knows>
<http://www.rmep.org/accounts/jojo10/>.
<http://www.rmep.org/accounts/chunkylover53/> <http://xmlns.com/foaf/0.1/knows>
<http://www.rmep.org/accounts/lisa1984/>.
```

In addition to those pure RDF serializations, it is also possible to embed RDF inside other documents such as XHTML pages to enrich them with semantic data. The W3C-supported way to do this is RDFa but other similar efforts are purposed like Microformats and HTML 5 Microdata. Below, you can see one way to express our sample graph as an XHTML page by leveraging RDFa special attributes.

```
<div xmlns:foaf=" http://xmlns.com/foaf/0.1/"
  about=" http://www.rmep.org/accounts/chunkylover53/">
  <span property="foaf:firstName">Homer</span>
  <span property="foaf:lastName">Simpson</span>
  <a rel="foaf:mbox" href="mailto:homer@simpson.org">
homer@simpson.org</span>
  <a rel="foaf:knows" href="http://www.rmep.org/accounts/bretzels34/">
Marge </a>
  <a rel="foaf:knows" href="http://www.rmep.org/accounts/jojo10/">
Bart</a>
  <a rel="foaf:knows" href=" http://www.rmep.org/accounts/lisa1984/">
Lisa</a>
</div>
```

The mixed approach illustrated by RDFa makes is easy to embed semantic data in web pages, but also makes it harder to extract that data back into proper RDF. To solve these issues, W3C proposed GRDDL as a standard way to extract the RDF data by applying XSLT stylesheets to the XHTML documents.

The Schema.org initiative

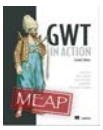
In June 2011, Google, Bing and Yahoo! search engines launched the Schema.org initiative to facilitate the semantic annotation of data in regular web pages. They provide a way to express semantic data using HTML Microdata including both common ontologies for things such as Persons, Places, Organizations, and so on and also support for their extraction in the most popular web search engines. Mapping to RDFa is also specified, making it a great option to consider for mixing RDF and web pages.

Summary

You've just seen how powerful the RDF data model is capable of modeling anything in a precise way and how flexible it can be used in various representation formats depending on the usage context.

Here are some other Manning titles you might be interested in:

Spring in Action, Third Edition
Craig Walls



GWT in Action, Second Edition
Adam Tacy, Robert Hanson, Jason Essington, Ian Bambury, and Christopher Ramsdale

Ruby in Practice
Jeremy McAnally and Assaf Arkin

Last updated: January 11, 2012