

Linked Data & Semantic Web

Monday, January 30, 2017 2:01 PM

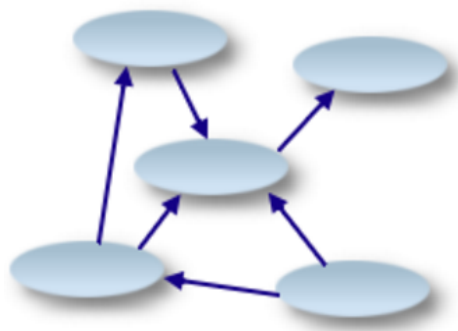
Semantic Web is a Web 3.0 web technology - a way of linking data between systems or entities that allows for rich, self-describing interrelations of data available across the globe on the web.

Semantic web seeks to change the landscape of the internet with regard this problem in a number of ways:

- Opening up the web of data to artificial intelligence processes (getting the web to do a bit of thinking for us).
- Encouraging companies, organizations and individuals to publish their data freely, in an open standard format.
- Encouraging businesses to use data already available on the web (data give/take).

Graph Data

RDF is a common acronym within the semantic web community because it forms one of the basic building blocks for forming the web of semantic data. What it defines is a type of database which you may not be immediately familiar with: something called a **graph database**.

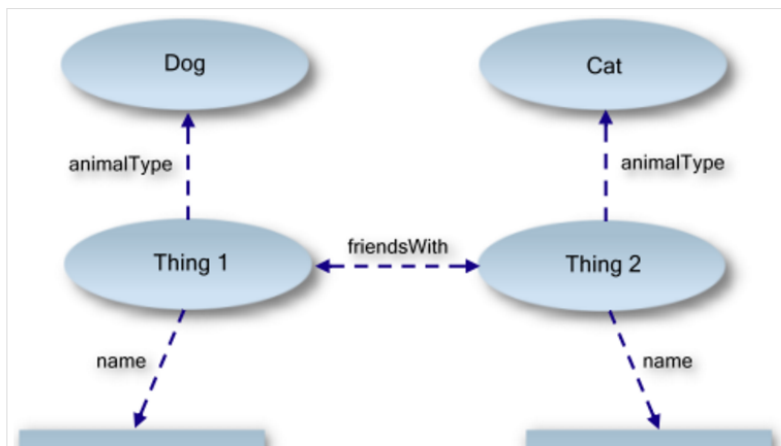


Graph DB
Arbitrary Object Relations
No Intrinsic Importance

In a data graph, there is no concept of roots (or a hierarchy). A graph consists of resources related to other resources, with no single resource having any particular intrinsic importance over another.

Bengie is a dog.
Bonnie is a cat.
Bengie and Bonnie are friends.

Using these three simple statements, let's turn this into a data graph:



Bengie

Bonnie

The relationships implied by this graph are fairly intuitive but to be thorough let's review them. We can see that our two *things* - identified by "Thing 1" and "Thing 2" - have the *properties* **name**, **animalType** and **friendsWith**.

RDF

If the graph data model is the model the semantic web uses to store data, RDF is the format in which it is written.

RDF (Resource Description Framework) format, and saw how it defined [statements](#) comprising a *subject*, a *predicate* (property), and an *object*. The **subject->predicate->object** relationship is called a *triple*. You also learned that RDF is the foundation upon which the web of semantic data is built.

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:feature="http://www.linkeddatatools.com/clothing-features#"
04.
05.   <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
06.
07.     <feature:size>12</feature:size>
08.     <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
09.
10.   </rdf:Description>
11.
12. </rdf:RDF>
```

You'll notice this one isn't quite the same as the last one. Whereas the last one had the *literal value* 12, this one is referring to the subject (ID) of another statement. That's right - objects in RDF can refer (reference) the subjects of other statements.



Remember A *subject* in an RDF document may also be referenced as a *object* of a property in another RDF statement (in the *resource* attribute). This can be a confusing concept for those starting out with RDF.

So this simply says "The subject has a property with name **feature:color** with object referring to the statement with ID **http://www.linkeddatatools.com/colors#white**".

```
1. <rdf:Description rdf:about="subject">
2.   <predicate rdf:resource="object" />
3.   <predicate>literal value</predicate>
4. </rdf:Description>
```

Here you see the *subject* of the statement (what the statement is about), and the two forms of predicates (*literal values* and *resources*, which reference other RDF statements).



Note The **rdf:Description** RDF/XML element allows you to group one or more statements into a single container. The above general form actually contains two statements referring to the same subject, but with two predicates and objects: a resource and a literal.

Unique IDs that we've been using so far such as **http://www.linkeddatatools.com/clothes#t-shirt** are called **Uniform Resource Identifiers**, or URIs for short. We've been using URIs to give a unique ID to the subjects, predicates or objects of statements so far without really saying why.

Because URIs are so crucial to the driving purpose behind RDF - to make data exchangeable globally - we will make a quick recap of URIs now. If you already understand URIs, you can skip this section.

XML Namespace URIs

Look back again at our example RDF document we've built. See that the T-shirt *size* predicate has the

name **feature:size** on line 07 below:

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:feature="http://www.linkeddatatools.com/clothing-features#"
04.
05.   <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
06.
07.     <feature:size>12</feature:size>
08.     <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
09.
10.   </rdf:Description>
11.
12. </rdf:RDF>
```

On line 03, notice that we've defined the *XML namespace* **feature** by giving it the *namespace URI* **http://www.linkeddatatools.com/clothing-features#**.

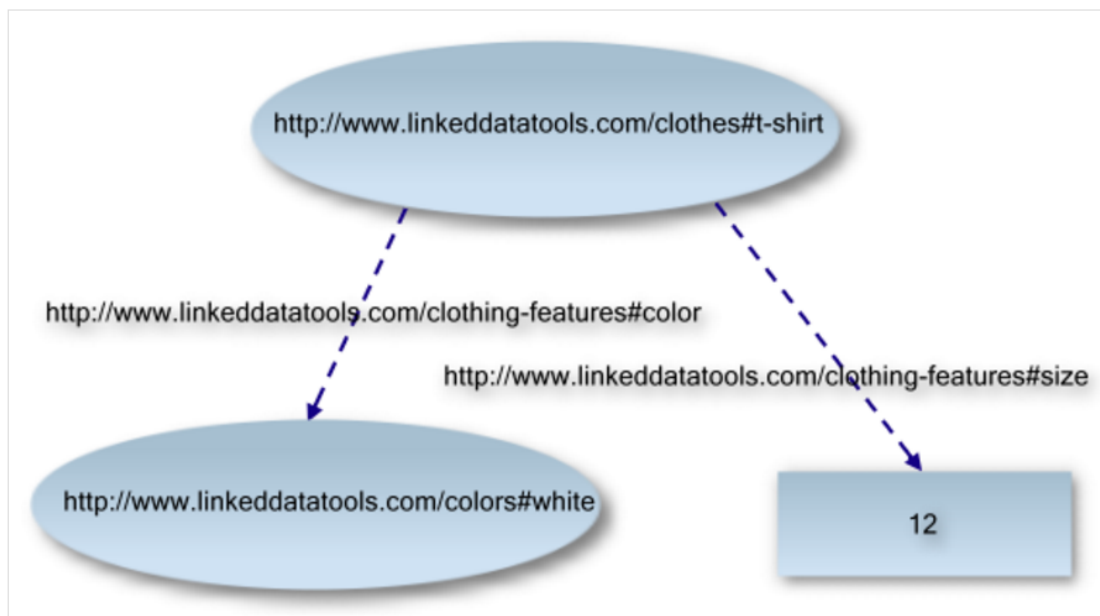
The purpose of this namespace is simply to avoid name conflicts with tags of the same name: other tags with the name "size" could be defined with other namespace URIs, and an RDF reader would still be able to tell that they were different properties even though they had the same tag name.

To get a fully qualified URI for **feature:size**, simply substitute the prefix **feature:** with its namespace URI, obtaining the full name **http://www.linkeddatatools.com/clothing-features#size**.



Note XML namespace URIs in RDF are used to distinguish between properties with the same (tag) name. To get the fully qualified URI, simply substitute the namespace prefix with the namespace URI.


Now we can state the graph with fully qualified URIs.






As you can see, it's not always easy or convenient to represent URIs in full in RDF graph diagrams. Often, shorthand versions are used instead (i.e. only using the namespace prefix).

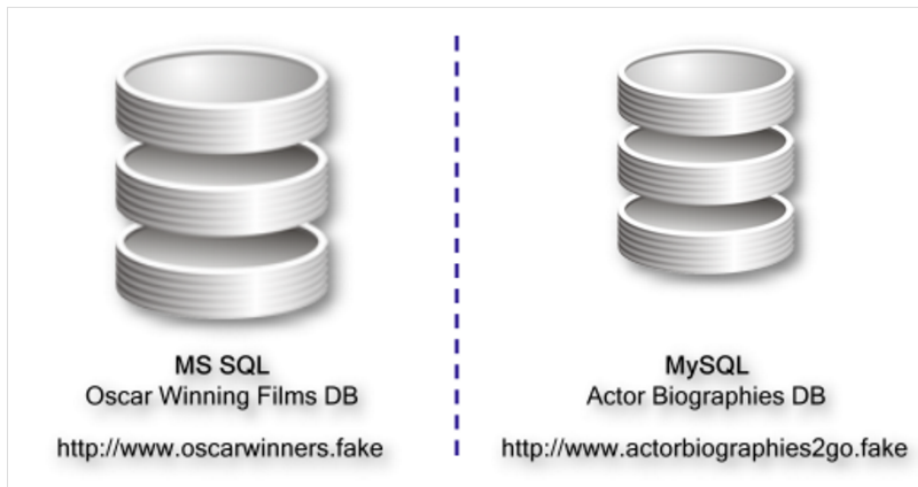
Semantic Modeling

Comparing the features of the mainstream ways of modeling data versus the semantic web model

Model	Example Format	Data	Metadata	Identifier	Query Syntax	Semantics (Meaning)
 Object Serialization	.NET CLR Object Serialization	Object Property Values	Object Property Names	e.g. Filename	LINQ	N/A

 Relational	MS SQL, Oracle, MySQL	Table Cell Values	Table Column Definitions	Primary Key (Data Column) Value	SQL	N/A
 Hierarchical	XML	Tag/Attribute Values	XSD/DTD	e.g. Unique Attribute Key Value	XPath	N/A
 Graph	RDF/XML, Turtle	RDF	RDFS/OWL	URI	SPARQL	Yes, using RDFS and OWL

Why include semantics in data ? Knowledge Integration Sharing Without Semantic Modeling



Our two sites, one fronting an MS SQL database of all Oscar winning films, and another one fronting a MySQL database of Hollywood actors, reside at <http://www.oscarwinners.fake> and <http://www.actorbiographies2go.fake> respectively. The two sites were started independently, and do not collaborate.

Let's look at how these two sites might collaborate under their current, more traditional data model:

- Obviously, the users of <http://www.oscarwinners.fake> would benefit from being able to click on the name of a starring actor and find out more about them - this information is stored in the MySQL database at <http://www.actorbiographies2go.fake>.
- Likewise, the users of <http://www.actorbiographies2go.fake> would benefit from being able to click on the names of films that the actors starred in and find more information. This is stored in the MS SQL database at <http://www.oscarwinners.fake>.
- Any sharing of data between the two sites cannot be done by joining tables in their databases. Firstly, they have been independently designed in the first place and so their primary keys referring to individual actors or films in both databases will not be synchronized. They would have to be mapped. But secondly, they are using different database server systems which are not cross-compatible.
- To collaborate using their current databases, the owners of either site would have to decide on a common data format by which to share information that they could both understand by using a common film and actor unique ID scheme of their own invention. They could do this, for example, by creating a secure XML endpoint on each of their websites from which they can request information from each other on demand. This way, their shared information is always up to date.



Important Point This sort of information interchange across incompatible, independently designed data systems takes time, money and human contextual interpretation of the different datasets. It also is restrictive to the data domains of only these two websites, any further additions to their knowledge from elsewhere will demand similar efforts. *It requires humans to understand the meaning of the data and agree on common formats to collaborate the two databases appropriately.*

RDFS and OWL

RDF data is annotated with semantic metadata using two principal syntaxes: RDFS and OWL. Both RDFS and OWL are [W3C specifications](#).

4.2 OWL Header

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/"
06.
07.   <!-- OWL Header Example -->
08.   <owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
09.     <dc:title>The LinkedDataTools.com Example Plant Ontology</dc:title>
10.     <dc:description>An example ontology written for the LinkedDataTools.com RDFS &
11.       OWL introduction tutorial</dc:description>
12.   </owl:Ontology>
13.
14.   <!-- Remainder Of Document Omitted For Brevity... -->
15. </rdf:RDF>
```

Although an ontology doesn't have to include a header, it is a good place to include information that will help others to understand what your ontology contains.

As above, we've included a title and description for the ontology. But, this is also the place where we could include version information (to make sure you can appropriately track and communicate updates to your ontology) and where you can state that your ontology imports other ontologies.

If your ontology does use elements from other ontologies, this will be absolutely necessary for tools or frameworks to know that your ontology is dependent on others.

OWL Classes, Subclasses & Individuals

The primary purpose of your ontology is to classify things in terms of semantics, or meaning. In OWL, this is achieved through the use of classes and subclasses, instances of which in OWL are called individuals. The individuals that are members of a given OWL class are called its class extension.

A class in OWL is a classification of individuals into groups which share common characteristics. If an individual is a member of a class, it tells a machine reader that it falls under the semantic classification given by the OWL class.

An Example

Here is our example OWL ontology again, this time with some added classes and subclasses. We define three plant classes: the **flowering plants** class and **shrubs** class. which are both subclasses of the **planttype** class.

The **planttype** class is the highest level class of all plant types.

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/"
06.   xmlns:plants="http://www.linkeddatatools.com/plants#"
07.
08.   <!-- OWL Header Omitted For Brevity -->
09.
10.   <!-- OWL Class Definition - Plant Type -->
11.   <owl:Class rdf:about="http://www.linkeddatatools.com/plants#planttype">
12.
13.     <rdfs:label>The plant type</rdfs:label>
14.     <rdfs:comment>The class of all plant types.</rdfs:comment>
15.
16.   </owl:Class>
17.
18.   <!-- OWL Subclass Definition - Flower -->
19.   <owl:Class rdf:about="http://www.linkeddatatools.com/plants#flowers">
20.
21.     <!-- Flowers is a subclassification of planttype -->
22.     <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
23.
24.     <rdfs:label>Flowering plants</rdfs:label>
25.     <rdfs:comment>Flowering plants, also known as angiosperms.</rdfs:comment>
26.
27.   </owl:Class>
28.
29.   <!-- OWL Subclass Definition - Shrub -->
30.   <owl:Class rdf:about="http://www.linkeddatatools.com/plants#shrubs">
31.
32.     <!-- Shrubs is a subclassification of planttype -->
33.     <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
```



```

34.     <rdfs:label>Shrubbery</rdfs:label>
35.     <rdfs:comment>Shrubs, a type of plant which branches from the base.
36.     </rdfs:comment>
37.
38. </owl:Class>
39.
40. <!-- Individual (Instance) Example RDF Statement -->
41. <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
42.
43.     <!-- Magnolia is a type (instance) of the flowers classification -->
44.     <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
45.
46. </rdf:Description>
47.
48. </rdf:RDF>

```

OWL Properties

Individuals in OWL are related by properties. There are two types of property in OWL:

- Object properties (owl:ObjectProperty) relates individuals (instances) of two OWL classes.
- Datatype properties (owl:DatatypeProperty) relates individuals (instances) of OWL classes to literal values.

An Example

Let's first add a data type property (one which links an instance to a literal value) and add the name of the species family the Magnolia is part of.

```

01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/"
06.   xmlns:plants="http://www.linkeddatatools.com/plants#">
07.
08.   <!-- OWL Header Omitted For Brevity -->
09.
10.   <!-- OWL Classes Omitted For Brevity -->
11.
12.   <!-- Define the family property -->
13.   <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>
14.
15.   <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
16.
17.     <!-- Magnolia is a type (instance) of the flowers class -->
18.     <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
19.
20.     <!-- The magnolia is part of the 'Magnoliaceae' family -->
21.     <plants:family>Magnoliaceae</plants:family>
22.
23.   </rdf:Description>
24.
25. </rdf:RDF>

```

Finally, let's add an *object* property (one which links an instance to another instance). Let's say we're running a shop, and we want to link this plant (Magnolia) to another plant which we know as the shop owner is equally as popular. Let's add a property called "similarlyPopularTo":

```

11.   <!-- Define the family property -->
12.   <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>
13.
14.   <!-- Define the similarlyPopularTo property -->
15.   <owl:ObjectProperty
16.     <rdf:about="http://www.linkeddatatools.com/plants#similarlyPopularTo"/>
17.
18.   <!-- Define the Orchid class instance -->
19.   <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#orchid">
20.
21.     <!-- Orchid is an individual (instance) of the flowers class -->
22.     <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
23.
24.     <!-- The orchid is part of the 'Orchidaceae' family -->
25.     <plants:family>Orchidaceae</plants:family>
26.
27.     <!-- The orchid is similarly popular to the magnolia -->
28.     <plants:similarlyPopularTo

```

```
28.     Ⓜ rdf:resource="http://www.linkeddatatools.com/plants#magnolia"/>
29. </rdf:Description>
30.
31. <!-- Define the Magnolia class instance -->
32. <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
33.
34.     <!-- Magnolia is an individual (instance) of the flowers class -->
35.     <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
36.
37.     <!-- The magnolia is part of the 'Magnoliaceae' family -->
38.     <plants:family>Magnoliaceae</plants:family>
39.
40.     <!-- The magnolia is similarly popular to the orchid -->
41.     <plants:similarlyPopularTo
42.     Ⓜ rdf:resource="http://www.linkeddatatools.com/plants#orchid"/>
43.
44. </rdf:Description>
45. </rdf:RDF>
```