Tutorial 2: Introducing RDF/XML

Next: Semantic Modeling

If the graph data model is the model the semantic web uses to store data, RDF is the format in which it is written. In the first lesson, we looked at graph data and introduced RDF. Now, we go further and describe the groundwork you need to write graph data yourself using RDF/XML - one of the most popular RDF formats on the web.

After this tutorial, you should be able to:

- Formally understand RDF statements, including objects as *resources* and *literals*.
- Write basic RDF documents in RDF/XML.
- Understand how RDF identifies resources using URIs (Universal Resource Identifiers).
- Understand why RDF is ideal for what it is built for: exchanging information, globally.

Estimated time: 5 minutes

You should have already understood the following tutorial (and pre-requisites) before you begin:

- Tutorial 1: Introducing Graph Data

In our first lesson, you looked at the concept of the graph database and made some comparisons with the more traditional forms of data storage: the *hierarchical* and *relational* data models.

We then looked briefly at 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.

During this lesson, you will learn step-by-step how to build your own RDF statements and understand them visually in a graph. This will then allow you to start thinking about adding semantics (meaning) to your data and understand the profound advantages this offers.

To do this best, let's build a simple RDF document, step-by-step.

2.1 Building An RDF Document

Add The RDF Document Root Tag

First, add the RDF root node:

```
1.  <rdf:RDF
2.      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
3.
4.      <!-- Body Code Omitted -->
5.
6.  </rdf:RDF>
```

If you look at line 02, you will see the standard W3.org namespace **http://www.w3.org/1999/02/22-rdf-syntax-ns#**. This namespace tells any machine reader that the enclosing document is an RDF document, and that the rdf:RDF tag resides in this namespace.

This namespace, and the RDF node, forms the root of all RDF documents.

Add A Statement

An RDF document can contain more than one statement. For simplicity, we'll only add one. Start by adding a an rdf:Description tag, which in RDF/XML can contain one or more statements about the same subject:

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

    <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">

        <!-- Statement Code Omitted -->

    </rdf:Description>

</rdf:RDF>
```

The rdf:Description tag simply means "I'm going to describe something (a *subject*) and I'm giving it the unique ID **http://www.linkeddatatools.com/clothes#t-shirt**".

Add Predicates

There's no point in saying you're going to describe something, give it a unique ID but then not describe anything about it. RDF statements describe the characteristics of their subjects using properties, or *predicates* in RDF terminology.

For simplicity, let's start by adding one predicate: the size of our T-shirt.

```
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.
09.     </rdf:Description>
10.
11. </rdf:RDF>
```

See line 07. This simply says "The subject has a property with name **feature:size** which has the literal value 12". In RDF terminology, this is a *statement*.

Finally, let's add one more predicate: the color of the T-shirt.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:feature="http://www.linkeddatatools.com/clothing-features#">

    <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">

        <feature:size>12</feature:size>
        <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>

    </rdf:Description>

</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**".

2.2 Breaking Down The Statement

Now we've looked at a simple example of an RDF document, let's formalize what we've learned and break the statement into its component parts:

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

2.3 A More Thorough Example

Let's return to the more complex example we gave in our last lesson on graph data:

```
<?xml version="1.0" encoding="UTF-8"?>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:region="http://www.country-regions.fake/">

    <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Oxford">
        <dc:title>Oxford</dc:title>
        <dc:coverage>Oxfordshire</dc:coverage>
        <dc:publisher>Wikipedia</dc:publisher>
        <region:population>10000</region:population>
        <region:principaltown rdf:resource="http://www.country-
        regions.fake/oxford"/>
    </rdf:Description>

</rdf:RDF>
```

To test your understanding and point out any areas where you need to recap, see if you can identify on the RDF document above the:

- **Subject** of the statement
- **Predicates** of the statement - including whether they are *resources* or *literals*
- **Objects** referenced by the resource predicates

Once you have understood RDF documents, and how they relate to data graphs, you are ready for our next lesson introducing the idea of modeling *semantics* into RDF graph data.

2.4 A Quick Recap Of URIs And XML Namespaces

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 exchangable 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>
```

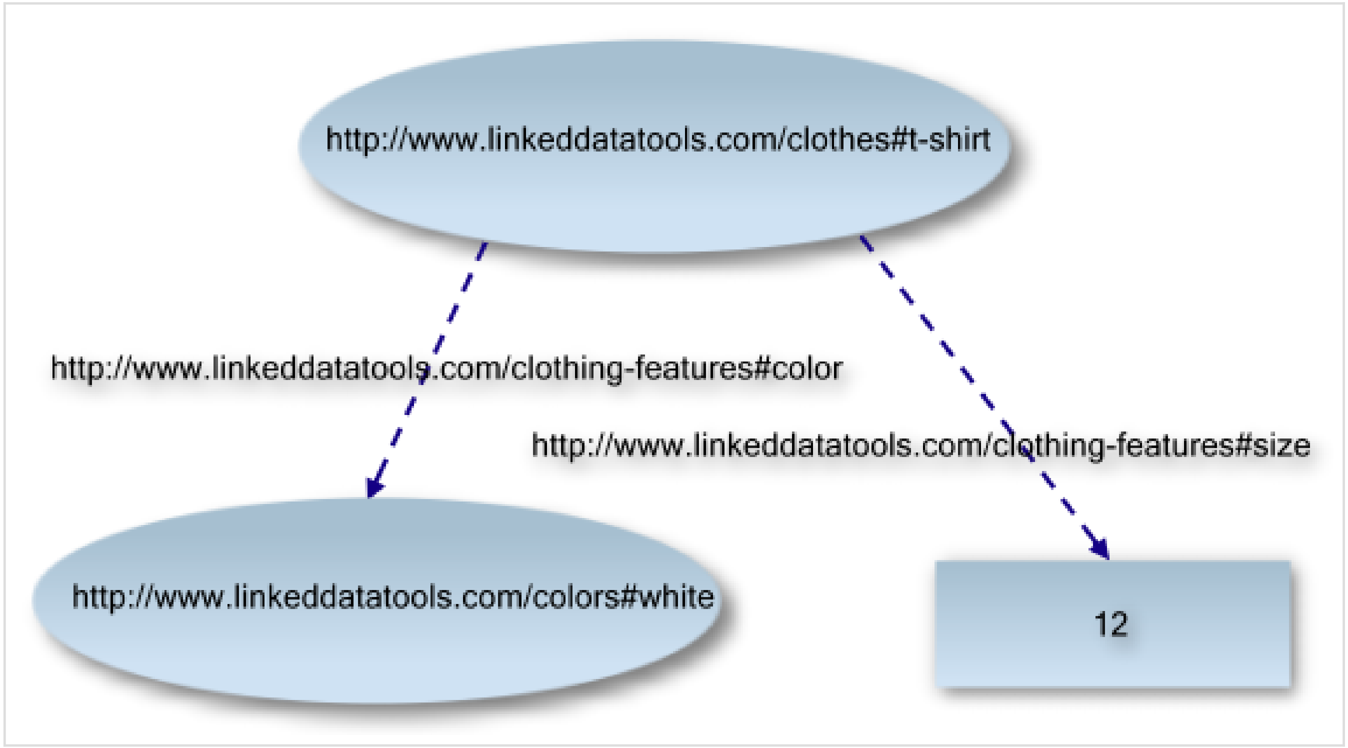
On line 03, notice that we've defined the *XML namespace* **feature** by giving it the *namespace URI* **http://www.linkeddatatools.com/clothing-features#**.

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.


http://www.linkeddatatools.com/clothes#t-shirt
http://www.linkeddatatools.com/clothing-features#color
http://www.linkeddatatools.com/clothing-features#size
http://www.linkeddatatools.com/colors#white
12

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).

You have completed this lesson. You should now understand the following:

- How to write your own basic RDF documents in RDF/XML
- Understand how and why RDF uses URIs to identify subjects, predicates and objects

- How to relate an RDF document to a corresponding data graph with fully qualified URIs

You should now be able to start the following tutorial:

- Tutorial 3: Semantic Modeling

SHARE

Comments

1 2 3

#29 2016-05-13 07:43
thnk u very much......
Quote

#28 2016-04-17 22:38
Thanksssssssssss sssssssssssssss s sooooooooo
Quote

#27 2016-03-30 08:55
Superb introduction about the Resource Description Framework.
Quote

#26 2016-01-26 17:34
Easily Understandable....:)
Quote

#25 2015-04-13 11:26
great
Quote

#24 2013-10-16 21:06
thank you very much for this tuto, very helpful

Quote

#23 **lorein** 2013-08-21 12:42

thanx for this tutorial but i want to ask what is the tool that i should download to write all those codes ?

Quote

#22 2013-07-09 09:40

this is awesome! I had a lecture about this and it was a pain in the a**!
now i read all this in 20min and everything is clear!
Thank you guys, you are great!
exams inc

Quote

#21 2013-07-01 00:40

Quoting #7:

> Very informative. even the text books that i went through dint have this crystal clear info that is provided in this tutorial. im happy i cam e across this. For a beginner like me to the Semantic Web this is the best tutorial start on.

Same for me

Quote

#20 2013-06-25 23:02

great tutorial thanks!

Quote

1 2 3

Refresh comments list

RSS feed for comments to this post.

Add comment

Name (required)

1000 symbols left

Refresh

Send

JComments

Community

Register to download software from our site and interact with other users as you learn semantic web.

A+ A- Reset