

Introduction to the semantic web  
or  
tripping with triples...

Pascal Mainini

pascal <at> impressionet <dot> ch

2008-07-05

# About me

Before we start - some words about myself...

- ▶ I'm Pascal Mainini
- ▶ I'm also known as gurke
- ▶ Open minded, critical hacker
- ▶ Started with computers almost 20 years ago, working professionally since nearly 10 years
- ▶ Currently network and security specialist
- ▶ *I don't like beeing photographed or recorded otherwise, thanks!*

# About this speech

This speech will

- ▶ Give an idea of what the semantic web is about
- ▶ Give an overview of underlying technologies
- ▶ Serve as a starting point for further explorations of the semantic web

# About this speech

This speech will **NOT**

- ▶ Provide an exact mathematical background (I'm too lame for that...)
- ▶ Give an in-depth tutorial of the technologies used (URIs, XML, RDF...)
- ▶ Allow you to start working with semweb-technologies without investing any further work

## About this speech

This speech and additional material is licensed under a  
“Attribution-Noncommercial-Share Alike 3.0 Unported” license:

<http://creativecommons.org/licenses/by-nc-sa/3.0/>

The slides and additional material can (soon) be found at:

<http://impressionet.ch/semwebspeech2>



# Prerequisites

To understand the examples in this speech, you will need a basic understanding of

- ▶ URIs
- ▶ XML and XML namespaces
- ▶ a bit of XML DTDs and schema

**If you have questions, just ask them right away during the speak!**

# Problems of the current web

The web as we know of it today has some problems. . .

- ▶ A gigantic bunch of information, a large diversity of formats
- ▶ This information is stored in a form understandable for humans (which is great!)
- ▶ It's not that easy for a machine to understand. . .
- ▶ Thus, information is hard to find and reuse

# Solution approaches

To solve these problems, two possible ways exist:

- ▶ Either, improve the usage of what's already there
  - ▶ By improving techniques, especially artificial intelligence
  - ▶ This is hard to accomplish and the results so far aren't satisfying
- ▶ Or by providing the information in a way better understandable by machines
  - ▶ This requires standardised formats
  - ▶ These must be formally correct, simple and easily extensible

# The semantic web

This is where the idea of the semantic web comes in.

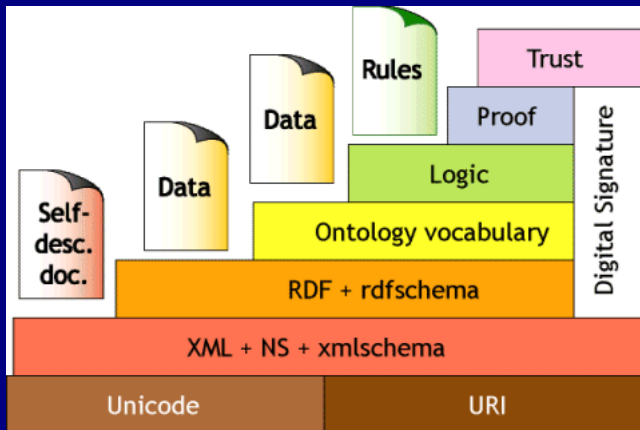
A full set of standards to accomplish this has been created by the W3C.

Technologies used base (not exclusively) on **XML** and **URIs** as discussed before.

On top of these follow **RDF**, **rdfschema** and **OWL**.

This is called the *semantic web layer cake*, let's have look at it. . .

# The layer cake



[w3.org]

This talk focuses on layers 3 and 4

# Syntax? Semantic?!?!?

While talking a lot about semantics and syntax. . .  
. . . what's this all about?

## *Syntax*

Syntax means defining rules about the structure of data, i.e. the structure of words in phrases or how well formed XML looks like.

## *Semantic*

Semantic in general defines the meanings of terms, sentences or XML- constructs.

For example, the word “*wine*” has different meanings while being syntactically the same. It can represent a beverage or also a famous piece of software.

# History

Important points in the history of the semantic web:

- ▶ Some initial work during 1997-1998
- ▶ In 1999
  - ▶ February: First recommendation, RDF model and syntax
  - ▶ March: rdfschema proposal
- ▶ February 2004: A suite of RDF and OWL recommendations, rdfschema recommendation
- ▶ Most widely used since for RSS-feeds (but not known for that. . .)
- ▶ My first contact with it: 2003

# Basic concepts

*“Everything should be representable, so one needs a common model with great generality”*

*“Two basic elements:*

*Assertions*

*Quotations (statements about assertions)”*

[<http://www.w3.org/DesignIssues/Semantic.html>]

# RDF model

This leads to a very simplistic model, to RDF:

- ▶ Information is represented as a *triple*, as a statement
- ▶ Every triple consists of three elements:
  - ▶ **Subject**
  - ▶ **Predicate**
  - ▶ **Object**
- ▶ Subjects and predicates are given as URIs
- ▶ Objects can either be other URIs or literal data
- ▶ RDF data is represented by directed graphs

## Example. . .

Based on that knowledge, we can build a first example.

Given the following natural language assumptions:

*“Mary has a lamb.”*

*“Mary is 14.”*

*“Big bad wolf wants this lamb.”*

*“Big bad wolf engages the seven dwarfs.”*

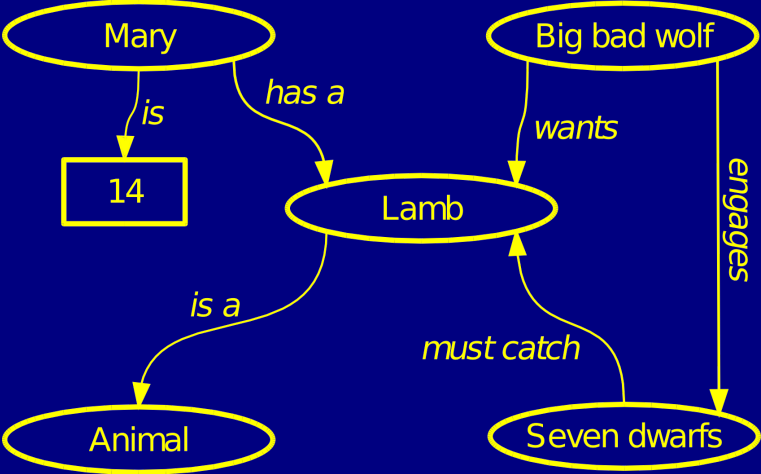
*“The seven dwarfs must steal the lamb.”*

. . . and of course. . .

*“the lamb is an animal!”*

Represented as a graph, this would look like. . .

# Example - graph



# Serialisation

Expressing triples as graphs is nice and comprehensible - for humans. But again - not very well parseable. Usually triples are handled in some serialized form. Multiple serialisation forms exist.

- ▶ **Notation 3 (N3)** - first form, quite complex
- ▶ **N-Triples** - subset of N3, possible recommendation
- ▶ **Turtle** - extension of N3
- ▶ **XML**

Turtle and XML are the most used forms. Turtle is quite practical when writing triples by hand, XML of course in automated exchange.

# Example - turtle serialisation 1

```
↳ <http://example.org/Mary> <http://example.org/has>
<http://example.org/Lamb> .
↳ <http://example.org/Mary> <http://example.org/is> ‘‘14’’ .
↳ <http://example.org/BigBadWolf> <http://example.org/wants>
<http://example.org/Lamb> .
↳ <http://example.org/BigBadWolf> <http://example.org/engages>
<http://example.org/SevenDwarfs> .
↳ <http://example.org/SevenDwarfs>
<http://example.org/muststeal> <http://example.org/Lamb> .
↳ <http://example.org/Lamb> <http://example.org/isa>
<http://example.org/Animal> .
```

*Note: lines are split up due to space. Each line starts with ↳*

## Example - turtle serialisation 2

Of course this isn't very handy, so here is a more cleaned up version:

```
↳ @prefix ex: <http://example.org> .  
↳ ex:Mary ex:has ex:Lamb ;  
↳ ex:is '14' .  
↳ ex:BigBadWolf ex:wants ex:Lamb ;  
↳ ex:engages ex:SevenDwarfs .  
↳ ex:SevenDwarfs ex:muststeal ex:Lamb .  
↳ ex:Lamb ex:isa ex:Animal
```

*Note: Turtle provides also other shortcuts not shown here*

## Example - XML serialisation

I will now show you this example serialised as XML.

Unfortunately, I didn't have the time to fully remake it, so i took the one from the previous speech - but there are only a few things missing and it will be enough to demonstrate how it looks like. . .

# RDF - additional features

RDF provides additional features for added expressiveness

- ▶ Typed literals: literals are identified by an URI, often used for XMLSchema-datatypes
- ▶ Localisation of literals (language specific literals)
- ▶ Empty nodes for complex relationships
- ▶ Containers, lists, collections

*I won't cover these in this talk, please refer to additional literature.*

## rdfschema - simple ontologies

rdfschema can be used to describe simple **ontologies**.

An ontology is a *network* of informations with logical relations. One can also think of it as “knowledge in a jar”, an ontology usually covers one or more areas of knowledge.

As opposed to this, a **taxonomy** is *hierarchically organised*. Example: In biology, the classification of a human as a mammal (with multiple levels inbetween of course), is a taxonomy.

## rdfschema - simple ontologies

rdfschema provides basic mechanisms for structuring RDF data.

Often, the functions given by rdfschema will be sufficient for simple ontologies.

Dublin core is a well known rdfschema ontology.

# Features of rdfschema

The most important constructs given by rdfschema are:

- ▶ `rdfs:Class`, `rdfs:subClassOf`
- ▶ `rdfs:Property`, `rdfs:subPropertyOf`, `rdfs:range`,  
`rdfs:domain`
- ▶ `rdfs:type`
- ▶ `rdfs:Container` (used for lists, sequences etc.)

## rdfschema - examples

Let's look at some examples (based on the tale of the lamb and the wolf...):

```
ex:Person rdfs:subClassOf <http://genome.org/human>
```

```
ex:Mary rdfs:type ex:Person
```

```
ex:is rdfs:subPropertyOf  
<http://older.net/ageproperty>
```

```
ex:is rdfs:range ex:Person
```

Of course, these are only a few and very simple examples - I hope you get the idea!

# OWL - introduction

**OWL** is the **W**eb **O**ntology **L**anguage

Yes, that would be **WOL** and not **OWL**! but...

- ▶ It's clear how to pronounce OWL...
- ▶ This acronym is great for making logos...
- ▶ OWLs are associated with wisdom...
- ▶ This makes up a great backstory...

[<http://lists.w3.org/Archives/Public/www-webont-wg/2001Dec/0169.html>]

A theory is also, that this comes from Winnie the Pooh, where the owl wasn't able to write her name correctly...;-)

# OWL - variants

There are three different variants of OWL:

## **OWL Full**

- ▶ Contains **OWL DL** and **OWL Lite**
- ▶ Very expressive
- ▶ Not decidable (that causes headaches to reasoners...)
- ▶ Not fully supported by software

## **OWL DL**

- ▶ Contains **OWL Lite**
- ▶ Decidable
- ▶ Nearly fully software supported

## **OWL Lite**

- ▶ Decidable
- ▶ Fully software supported
- ▶ Less expressive

# OWL Lite - constructs

OWL Lite provides constructs for

- ▶ (In-)Equality
- ▶ Property characteristics
- ▶ Property restrictions
- ▶ Cardinality restrictions
- ▶ Header informations
- ▶ Class intersections
- ▶ Versioning
- ▶ Annotation
- ▶ Datatypes

And of course, rdfschema can also be combined with OWL...!

# OWL Lite - constructs

Some example of OWL Lite constructs:

- ▶ `owl:equivalentClass`, `owl:equivalentProperty`
- ▶ `owl:ObjectProperty`, `owl:DatatypeProperty`
- ▶ `owl:minCardinality`, `owl:maxCardinality`
- ▶ `owl:intersectionOf`
- ▶ `owl:versionInfo`
- ▶ `owl:AnnotationProperty`, `owl:OntologyProperty`

# OWL DL/Full - constructs

Additionally, OWL DL and OWL Full introduce the following constructs:

- ▶ `oneOf`
- ▶ `dataRange`
- ▶ `disjointWith`
- ▶ `equivalentClass`
- ▶ `unionOf`, `complementOf`
- ▶ `maxCardinality`, `cardinality`
- ▶ `hasValue`

# Full example

Let's have a look at a simple example which shows what has been discussed until now.

This example has been taken from wikipedia.

# What can be done now?

We've learned a lot about technologies so far - but how can this be used?

There are a lot of APIs out there for various languages, which help working with semantical information and ontologies. Examples include Jena (JAVA) and LibRDF (multiple languages).

Also, there is a multitude of triplestores. Triplestores are the databases for RDF-data, they store - as the name implies - triples.

*Check out my links for more information...!*

# Querying

RDF data can also be queried. Recently, **SPARQL** has been defined as an official standard.

Giving a broad introduction into SPARQL doesn't fit into the timeframe of this speech. I recommend you further reading on the web. . .

# Infering and Reasoning

Besides of that, also infering and reasoning are interesting and important applications of RDF.

You can understand those as making automatic assumptions about triples. As a (very simple) example, look at this:

*When we know that all human beeings are born. . .*

*. . . and we know that Pascal Mainini is a human beeing. . .*

*. . . we can automatically infer that Pascal Mainini has been born!*

Reasoning and infering go into artificial intelligence. I won't go any further here too - but it's a very interesting field and you can - again find a lot of information and tools on the web!

# Conclusion

We are reaching the end of this speech. . .  
I hope that:

- ▶ This was interesting to you. . . !
- ▶ I was able to give you a good overview over this broad topic. . . !
- ▶ You will be able to start your own explorations - if you like to. . . !

# Questions

Are there any...

... Questions?!?

Thanks!

Thanks a lot for your interest!

Check out

<http://impressionet.ch/semwebspeech2>

by the end of next week to find all the information!