Semantic Services

Michael Wollowski

Overview

- Motivation
- Web services
- RDF
- RDFS/OWL
- Inference
- Seamless web

Motivation

- Consider the future scenario of a wired world described in the next three slides.
- How many and what kind of databases did we access?

In Search of a Juicy Burger

A guy, let's call him Bob, has high cholesterol. His doctor strictly prohibits him from eating foods high in cholesterol. One day, while Bob goes about his daily business, he starts to have this nagging desire for a juicy hamburger. After much agonizing, he cannot resist his urges any longer and drives over to the nearest Burger King. Not one who is prone to exercise, Bob cruises down the drive-through lane to order a burger with fries. In the process of wirelessly deducting the cost of the meal from Bob's E-Z Pass account, Bob's computer realizes that the bill comes from a fast food establishment. The computer quickly studies the bill to notice that the order is for a hamburger and fries and promptly refuses payment and the order is cancelled. Bob's health insurance company made him install a piece of software which monitors his purchases of high-cholesterol foods. Muttering to himself inaudibly, Bob drives home.

In Search of a Juicy Burger

Pulling into the garage, he suddenly has a bright idea; let's take the old Triumph TR 6 out for a spin. It doesn't have an E-Z Pass system! He heads to the local burger joint, where cash is still accepted. Waiting in line and chatting to the people around him, he suddenly gets shoed away by an off-duty paramedic. The paramedic received an alarm from a centralized system which was able to pinpoint Bob's location based on the health monitor implanted in his body. The health monitor includes a GPS system so as to alert paramedics to his location, in case of emergency. Bob's location was pinpointed to be in a high-cholesterol zone, alerting the medic. After receiving an admonition from the medic, Bob heads home again.

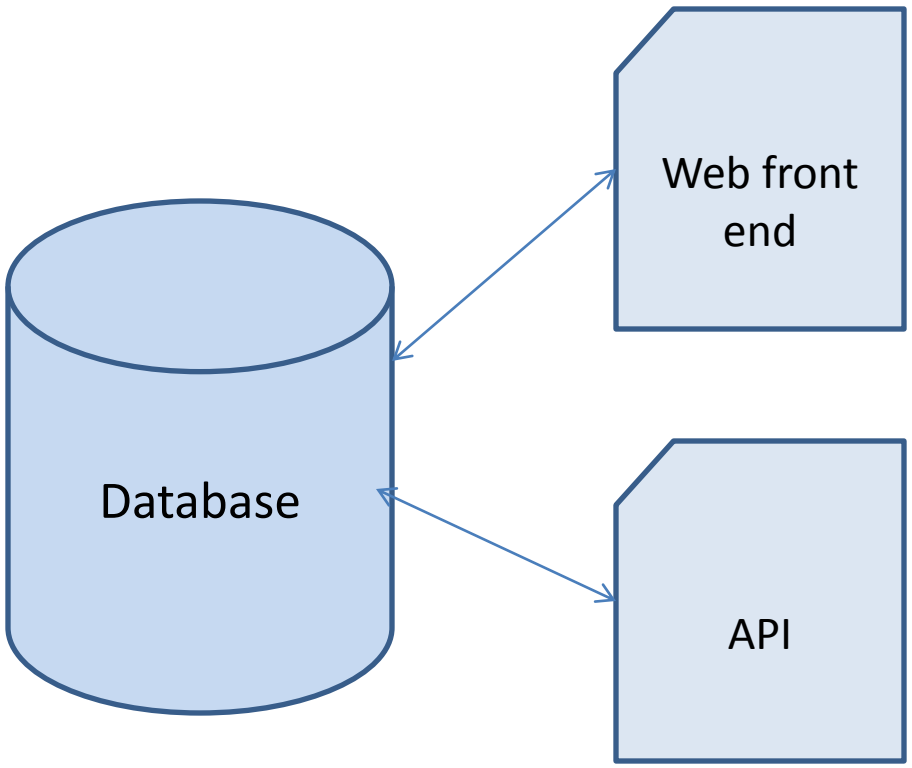
In Search of a Juicy Burger

His friendly neighbor, noticing him all dejected, shouts: “Hey Neighbor! Have a beer, I just turned on the grill, interested in joining me for a cookout?” Teary eyed and weak-legged, Bob agrees. While the hamburgers are sizzling on the grill, an alarm goes off in the central system. Bob had to agree that as part of a lower rate on his health insurance, the insurance company would get access to the images which his digital camera takes of his point of view every few seconds. The advanced image analysis software recognized flames, smoke, and juicy hamburgers on the grill. This time however, no one will keep Bob from his hamburger, as the personnel monitoring the alarms are socializing outside, over tofu burgers.

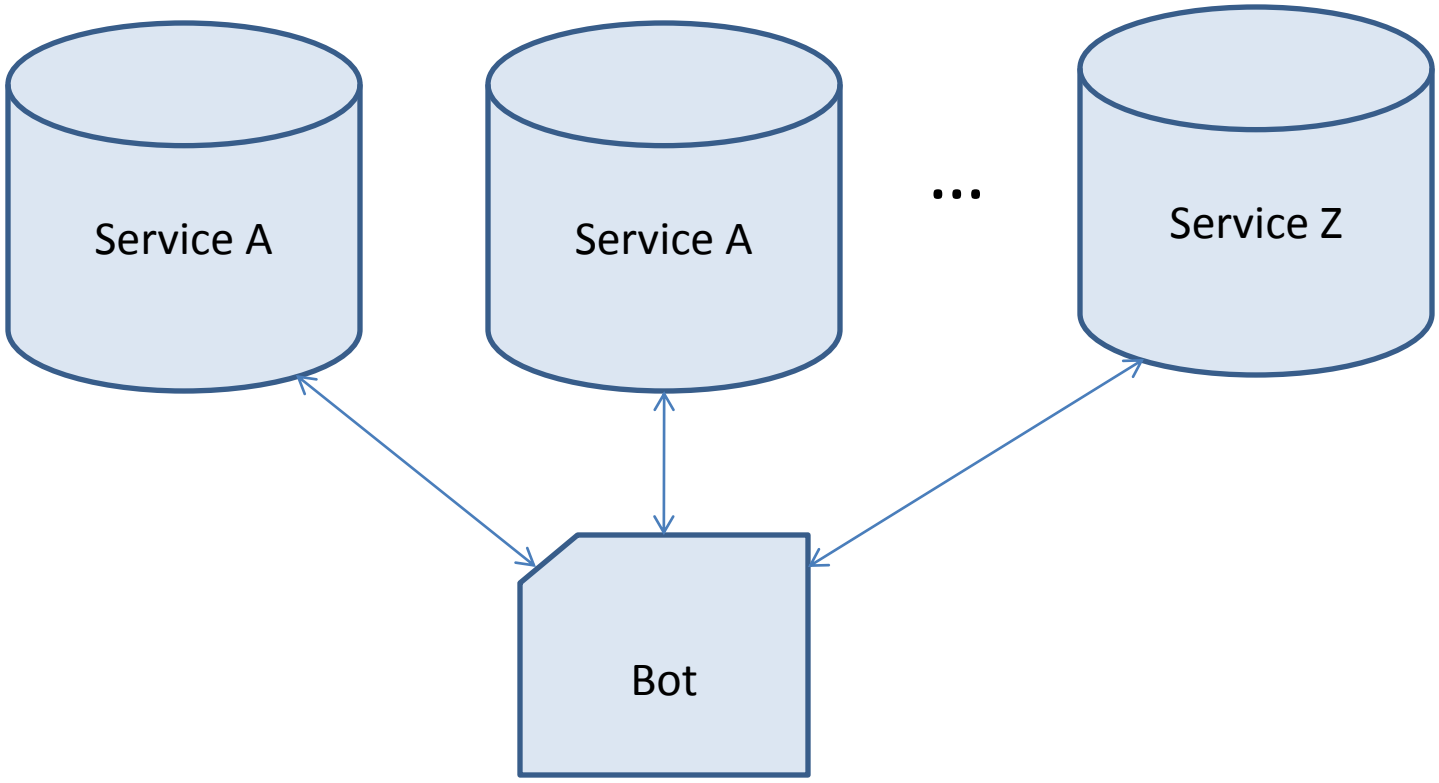
Web Services

- **Websites** like Autotrader.com are a database with a web-based GUI.
- A **web-service** is a database with an API.
- An application is meant to be see by people.
- A service is meant to be seen by software.
- A service acts behind the scene of the human readable web to accomplish tasks.

Web Services



Towards the Semantic Web



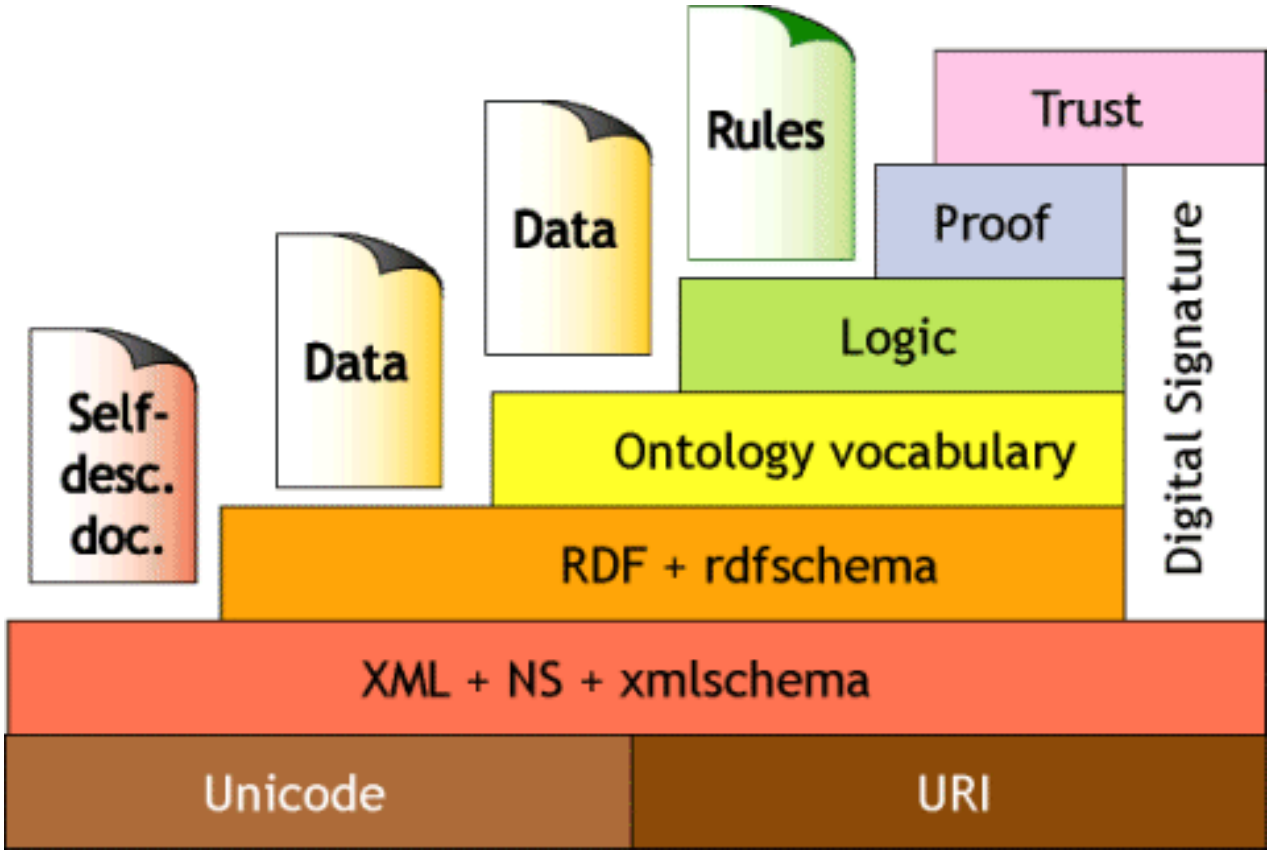
Exporting to XML

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988
...					

- <cd>
  <title>Empire Burlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  ...
  </cd>

XML Dialects

Solution Proposal: Semantics



RDF Statements

- A **Statement** consists of a *subject*, *predicate* and *object*
- Example:

The author of http://www.w3schools.com/RDF is Jan Egil Refsnes

- Subject: http://www.w3schools.com/RDF
- Predicate: author
- Object: Jan Egil Refsnes

RDF Statements

- In technical terms, we have a **Resource**, a **Property**, and a **Property value** corresponding to subject, predicate and object of a Statement.
- Example:

  The author of http://www.w3schools.com/RDF is Jan Egil Refsnes.
- Resource: http://www.w3schools.com/RDF
- Property: author
- Property value: Jan Egil Refsnes

Resources

- We can think of a resource as an object, a "thing" we want to talk about
  - E.g. authors, books, publishers, places, people, hotels
- Every resource has a **URI**, a Universal Resource Identifier
- A URI can be
  - a URL (Web address) or
  - some other kind of unique identifier

Properties

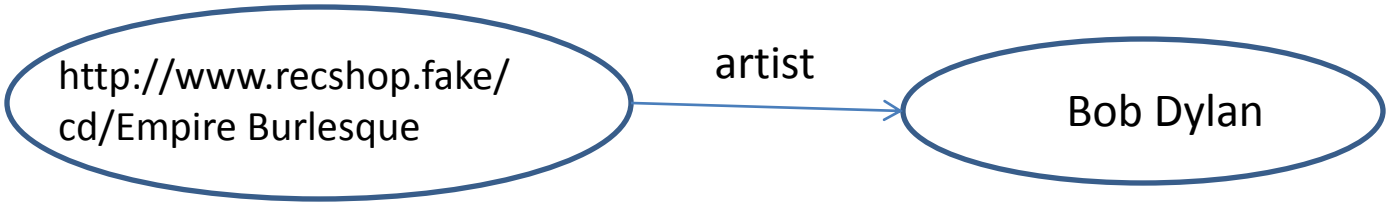
- Properties are a special kind of resources
- They describe relations between resources
  - E.g. “written by”, “age”, “title”, etc.
- Properties are also identified by URIs
- Advantages of using URIs:
  - A global, worldwide, unique naming scheme
  - Reduces the homonym problem of distributed data representation

RDF Example

- XML based

```
<?xml version="1.0"?>
<rdf:RDF >
<rdf:Description rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
    <cd:artist>Bob Dylan</cd:artist>
    <cd:country>USA</cd:country>
    …
</rdf:RDF>
```

Semantic Net



- A directed graph with labeled nodes and arcs
  - **from** the resource (the **subject** of the statement)
  - **to** the value (the **object** of the statement)
- Known in AI as a *semantic net*
- The value of a statement may be a resource
  - It may be linked to other resources

Example Evaluated

- How to avoid the Tower of Babel?
  - All services use same namespace
  - May as well agree on the same syntax for XML
  - In other words, develop a single standard
  - At least, develop a very small set of standards
- Alternatively, facilitate matching of namespace elements.

RDFS

- RDFS stands for RDF Schema
- It enables the definition of classes and class hierarchies
- Additionally, we can define property hierarchies

RDFS Example

```
<rdfs:Class rdf:ID=“recordingMedium" />

<rdfs:Class rdf:ID=“cd">
 <rdfs:subClassOf rdf:resource="#recordingMedium"/>
</rdfs:Class>

<rdfs:Property rdf:ID=“artist”/>

<rdfs:Property rdf:ID=“starvingArtist">
 <rdfs:subPropertyOf rdf:resource="#artist"/>
</rdfs:Class>
```

OWL

- More sophisticated relationships
- Disjointness of classes
  - Sometimes we wish to say that classes are disjoint (e.g. **male** and **female**)
- Boolean combinations of classes
  - Sometimes we wish to build new classes by combining other classes using union, intersection, and complement
  - E.g. **person** is the disjoint union of the classes **male** and **female**

OWL

- Cardinality restrictions
  - E.g. a person has exactly two parents, a course is taught by at least one lecturer
- Special characteristics of properties
  - Transitive property (like “greater than”)
  - Unique property (like “is mother of”)
  - A property is the inverse of another property (like “eats” and “is eaten by”)

Owl Example

```
<owl:Ontology rdf:about="xml:base"/>

<owl:Class rdf:ID="animal">
    <rdfs:comment>Animals form a class</rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="plant">
    <rdfs:comment>Plants are disjoint from animals. </rdfs:comment>
    <owl:disjointWith rdf:resource="#animal"/>
</owl:Class>

<owl:Class rdf:ID="tree">
    <rdfs:comment>Trees are a type of plant. </rdfs:comment>
    <rdfs:subClassOf rdf:resource="#plant"/>
</owl:Class>
```

Owl Example

```
<owl:Class rdf:ID="branch">
        <rdfs:comment>Branches are parts of trees. </rdfs:comment>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#is-part-of"/>
                        <owl:allValuesFrom rdf:resource="#tree"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="leaf">
        <rdfs:comment>Leaves are parts of branches. </rdfs:comment>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#is-part-of"/>
                        <owl:allValuesFrom rdf:resource="#branch"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Owl Example

```
<owl:Class rdf:ID="carnivore">
    <rdfs:comment>Carnivores are exactly those animals
    that eat also animals.</rdfs:comment>
    <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#animal"/>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#eats"/>
            <owl:someValuesFrom
                rdf:resource="#animal"/>
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

Owl Example

```
<owl:Class rdf:ID="lion">
    <rdfs:comment>Lions are animals that eat
    only herbivores.</rdfs:comment>
    <rdfs:subClassOf rdf:type="#carnivore"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#eats"/>
            <owl:allValuesFrom
                rdf:resource="#herbivore"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Owl Example

<lion rdf:ID=“22789" />

<lion rdf:ID=“Simba" />

Inference

- The richer the language is, the more inefficient the reasoning support becomes
- Sometimes it crosses the border of *noncomputability*
- We need a compromise:
  - A language supported by reasonably efficient reasoners
  - A language that can express large classes of ontologies and knowledge.

Inference

- Class membership
  - If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D
- Equivalence of classes
  - If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C, too

Inference

- Consistency
  - X instance of classes A and B, but A and B are disjoint
  - This is an indication of an error in the ontology
- Classification
  - Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an instance of A

Inference

- Many reasoners
- Few that work
- Jena (Java based)
- Some python libraries that would facilitate building a reasoner

Evaluation of Semantic Services

- A lot of technology
- Few semantic web applications
- Applications are typically an ontology on top of a web service
- Reasoning is expensive
- We have not covered key technology such as: service discovery, matching and execution

Seamless Web

- What we want to accomplish is what I would call a *seamless web*
- I believe it requires engineered solutions
- The less semantics, the more reliable
- The more reliable, the more satisfying to use

Student Work

- JP Verkamp and Brenon Smith

  FlickrFM
- Nic Hudson and Christine Price

  Restaurant Bot
- Bobby Bennett and Isaac Heyveld

  Shopping for Electronics Bot

References

- Book: A Semantic Web Primer, by Antoniou & van Harmelen
- Website for book: http://www.ics.forth.gr/isl/swprimer/
- RDF/S Tutorial: http://www.w3schools.com/rdf/default.asp
- OWL Guide: http://www.w3.org/TR/owl-guide/
- OWL Reference: http://www.w3.org/TR/owl-ref/
- Jena: http://jena.sourceforge.net/