An Introduction to Web Semantic Technologies

Resource Description Framework
RDF

Resource Description Framework (RDF)

- A framework for describing resources on the web
- What is a resource?
  - The answer is still the subject of debate.
  - For our purposes we can think of it as anything we can identify: you are a resource, as is your home page, this presentation, a book, song, student, etc..
- Designed to be read and understood by computers
- Not designed for being displayed to people
- Written in XML
- Part of the W3C's Semantic Web Activity
- A W3C Recommendation: http://www.w3.org/RDF
RDF Components

- Formal data model
- Syntax for interchange of data
- Schema Type system (schema model)
- Syntax for machine-understandable schemas
- Query and profile protocols

RDF - Examples of Use

- We can describe things such as:
  - Properties for shopping items, such as price and availability
  - Time schedules for web events
  - Information about web pages (content, author, created and modified date)
  - Content and rating for web pictures
  - Content for search engines
  - Describing electronic libraries
RDF and "The Semantic Web"

- The RDF language is a part of the W3C's Semantic Web Activity. W3C's "Semantic Web Vision" is a future where:
  - Web information has exact meaning
  - Web information can be understood and processed by computers
  - Computers can integrate information from the web

RDF: Basic Concepts

- The fundamental concepts of RDF are resources, properties, and statements.
- **Resources:**
  - An object, a “thing” we want to talk about.
  - Authors, books, publishers, places, people, hotels, rooms, search queries, and so on.
  - **Every resource has a URI**, a Uniform Resource Identifier. A URI can be a URL (Uniform Resource Locator, or Web address) or some other kind of unique identifier; note that an identifier does not necessarily enable access to a resource
  - URI schemes have been defined not only for Web locations but also for such diverse objects as telephone numbers, ISBN numbers, and geographic locations.
RDF: Basic Concepts (Cont..)

- **Properties:**
  - Special kind of resources
  - Describe relations between resources, for example “written by”, “age”, “title”, and so on.
  - Are also identified by URIs (and in practice by URLs). The use of such a scheme greatly reduces the homonym problem.

---

RDF: Basic Concepts (Cont..)

- **Statements:**
  - Assert the properties of resources.
  - Are a **object-attribute-value triple**, consisting of a resource, a property, and a value.
  - Values can either be resources or **literals**. Literals are atomic values (strings).
Three Views of a Statement

An example of a statement is

David Billington is the owner of the Web page http://www.cit.gu.edu.au/~db.

The simplest way of interpreting this statement is to use the definition and consider the triple:


We can think of this triple \((x, P, y)\) as a logical formula \(P(x, y)\), where the binary predicate \(P\) relates the object \(x\) to the object \(y\).

Three Views of a Statement (Cont..)

A second view is graph-based.

- It is a directed graph with labeled nodes and arcs.
- the arcs are directed from the resource (the subject of the statement) to the value (the object of the statement).
- This kind of graph is known in the Artificial Intelligence community as a semantic net.
Three Views of a Statement (Cont..)

As we have said, the value of a statement may be a resource. Therefore, it may be linked to other resources. Consider the following triples:

\[(\text{http://www.cit.gu.edu.au/~db}, \text{http://www.mydomain.org/site-owner}, \text{#DavidBillington})\]

\[(\text{#DavidBillington}, \text{http://www.mydomain.org/phone}, \text{"3875507"})\]


Three Views of a Statement (Cont..)

- A **third representation** of a Statement is possible based on **XML**.
- In this context, an **RDF document** is represented by an XML element with the tag **rdf:RDF**.
- The **content of this element** is a number of **descriptions**, which use **rdf:Description** tags.
- Every description makes a statement about a resource, which is identified in one of three different ways:
  - An **rdf:about** attribute, referencing an existing resource.
  - An **rdf:ID** attribute, creating a new resource.
  - **Without a name**, creating an anonymous resource.
Three Views of a Statement (Cont..)

Representing a statement about the resource http://www.cit.gu.edu.au/~db in XML

```xml
<?xml version="1.0" encoding="UTF-16"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:mydomain="http://www.mydomain.org/my-rdf-ns">
    <rdf:Description rdf:about="http://www.cit.gu.edu.au/~db">
        <mydomain:site-owner rdf:resource="#DavidBillington"/>
    </rdf:Description>
</rdf:RDF>
```

RDF Resource, Property, and Property Value

- Recalling that RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values.
- What are Resource, Property, and Property value?
  - A **Resource** is anything that can have a URI, such as "http://www.w3schools.com/rdf"
  - A **Property** is a Resource that has a name, such as "author" or "homepage"
  - A **Property value** is the value of a Property, such as "Juan Pérez" or "http://www.w3schools.com" (note that a property value can be another resource)
RDF Statements

- The combination of a Resource, a Property, and a Property value forms a Statement (known as the **subject**, **predicate** and **object** of a Statement).
- These Statements are also known as **Triples**

(<http://...isbn...6682>, <http://.../original>, <http://...isbn...409X>)

---

RDF triples (cont.)

- An RDF Triple (s,p,o) is such that:
  - “s”, “p” are URI-s, ie, resources on the Web; “o” is a URI or a literal
    - Here is a complete triple:

Example, Statement:

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

The **subject** of the statement is: **http://www.w3schools.com/rdf**

The **predicate** (or property) is: **author**

The **object** is: **Jan Egil Refsnes**

Statement:

"The homepage of http://www.w3schools.com/rdf is http://www.w3schools.com".

The **subject** of the statement above is: **http://www.w3schools.com/rdf**

The **predicate** is: **homepage**

The **object** is: **http://www.w3schools.com**

- RDF is a general model for such triples
  - with machine readable formats like **RDF/XML**, **Turtle**, **N3**, **RDFa**, **...**
RDF Data Model

- Directed labeled graphs
- Model elements
  - Resource
  - Property
  - Property value
  - Statement

RDF Model Primitives

```
Resource -> Value

Property

Statement
```
RDF Model Example

page.html

Creator

John Smith

Title

John’s Home Page

Basic Example

page.html

DC:Creator

John Smith

DC:Title

John’s Home Page
Basic RDF Example

<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
    xmlns:dc="http://purl.org/metadata/dublin_core#">
    <rdf:Description about="page.html">
        <dc:Creator>John Smith</dc:Creator>
        <dc:Title>John’s Home Page</dc:Title>
    </rdf:Description>
</rdf:RDF>

Structured Value

page.html Creator
  Title
    John’s Home Page
  Name
    John Smith
  Email
    js@corp.com
Structured Value Example

```
<?xml version="1.0"?>
<RDF
    xmlns=http://www.w3.org/TR/WD-rdf-syntax#
    xmlns:dc=http://purl.org/metadata/dublin_core#
    xmlns:corp=http://mycorp.com/corpSchema/>
    ...
```
Structured Value Example - 2

...<Description about="page.html">  
  <dc:Creator>  
    <Description>  
      <corp:Name>John Smith</corp:Name>  
      <corp:Email>js@corp.com</corp:Email>  
    </Description>  
  </dc:Creator>  
  <dc:Title>John’s Home Page</dc:Title>  
</Description>
</RDF>

Abbreviations: 1

<rdf:RDF>  
  <rdf:Description about="page.html">  
    <dc:Creator>  
      <rdf:Description  
        corp:Name="John Smith"  
        corp:Email="js@corp.com"  
      />  
    </dc:Creator>  
    <dc:Title>John’s Home Page</dc:Title>  
  </rdf:Description>  
</rdf:RDF>
Abbreviations: 2

```
<rdf:RDF>
  <rdf:Description about="page.html">
    <dc:Creator
corp:Name="John Smith"
corp:Email="js@corp.com"/>
    <dc:Title>John's Home Page</dc:Title>
  </rdf:Description>
</rdf:RDF>
```

RDF:ABOUT/RDF:RESOURCE/RDF:ID

How do we introduce subject and object URIs in RDF/XML?

- `rdf:about` sets the subject URI of a statement, which may be absolute (http://example.com/) or resolved relative to the BASE of the document (e.g. /foo/bar, #frag). (Like href in html):
  ```xml
  <?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/"
    xml:base="http://spam.com/foo/
  >
  ```

- `rdf:resource` sets the object URI of a statement, once again either absolute or relative.

- `rdf:ID` sets the subject URI, but it can only be within this document. An ID can also only be used once. Very like <a name="anchor"> or id="anchor" in html.
  rdf:ID is discouraged since you can replace it with an rdf:about or rdf:resource with a fragment #anchor and it can cause xml issues if you use the same id more than once. That is, it's redundant and a potential source of errors.
Reification

- In RDF it is possible to make statements about statements.
- Example: Grigoris believes that David Billington is the creator of the Web page http://www.cit.gu.edu.au/~db.
- This kind of statement can be used to describe belief or trust in other statements, which is important in some kinds of applications.
- The solution is to assign a unique identifier to each statement, which can be used to refer to the statement.
- RDF allows this using a reification mechanism.

Reification (Cont..)

Example: Let´s take the next Description

```
<rdf:Description rdf:about="949352">
  <uni:name>Grigoris Antoniou</uni:name>
</rdf:Description>
```

We can give an ID (reify) to the previous statement:

```
<rdf:Statement rdf:about="StatementAbout949352">
  <rdf:subject rdf:resource="949352"/>
  <rdf:object>Grigoris Antoniou</rdf:object>
</rdf:Statement>
```
Meta-description

- **page.html**
  - **Cost**: $0.15
  - **ValidUntil**: 18-Apr-2013

Property Reification

- **page.html**
  - **Cost**: $0.15
  - **Subject**
  - **Object**
  - **type**
  - **Predicate**
  - **ValidUntil**: 18-Apr-2013
Meta-description Example - 1

```xml
<?xml version="1.0"?>
<RDF
   xmlns:rdf
   xmlns:corp
   ="http://www.w3.org/TR/WD-rdf-syntax#"
xmlns:corp
   ="http://mycorp.com/schema/">
   ...
</RDF>
```

Meta-description Example - 2

```xml
   ...
   <rdf:Description about="page.html"
   bagID="D_001">
      <corp:Cost>$0.15</corp:Cost>
   </rdf:Description>
   <rdf:Description aboutEach="#D_001">
      <corp:ValidUntil>18-Apr-2013</corp:ValidUntil>
   </rdf:Description>
</rdf:RDF>
```
Meta-description Example - 3

```
<rdf:RDF>
  <rdf:Description about="page.html"
    bagID="D_001"
    corp:Cost="$0.15"/>
  <rdf:Description aboutEach="#D_001"
    corp:ValidUntil="18-Apr-1999"/>
</rdf:RDF>
```

---

Data Types

- Programs need to know how to interpret a resource
- Example, how to know that a literal represents an age:

```
(#DavidBillington, http://www.mydomain.org/age,
"27"^^http://www.w3.org/2001/XMLSchema#integer )
```

To define 2 categories (course and lecturer):

```
<rdf:Description rdf:about="CIT1111">
  <rdf:type rdf:resource="&uni;course"/>
  <uni:courseName>Discrete Mathematics</uni:courseName>
  <uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>

<rdf:Description rdf:about="949318">
  <rdf:type rdf:resource="&uni;lecturer"/>
  <uni:name>David Billington</uni:name>
  <uni:title>Associate Professor</uni:title>
</rdf:Description>
```
A simple RDF example (in RDF/XML)

(Note: namespaces are used to simplify the URI-s)

A simple RDF example (in Turtle)
A simple RDF example (in RDFa)

The book entitled "Le palais des miroirs" is the French translation of the "Glass Palace".

"Internal" nodes

- Consider the following statement:
  - “the publisher is a «thing» that has a name and an address (city)"
- Until now, nodes were identified with a URI. But...
- ...what is the URI of «thing»?
One solution: create an extra URI

- The resource will be “visible” on the Web
- care should be taken to define unique URI-s

```xml
<rdf:Description rdf:about="http://.../isbn/000651409X">
  <a:publisher rdf:resource="urn:uuid:f60ffe40-307d-..."/>
</rdf:Description>
<rdf:Description rdf:about="urn:uuid:f60ffe40-307d-...">
  <a:p_name>HarpersCollins</a:p_name>
  <a:city>HarpersCollins</a:city>
</rdf:Description>
```

*Internal* identifier ("blank nodes")

```xml
<rdf:Description rdf:about="http://.../isbn/000651409X">
  <a:publisher rdf:nodeID="A234"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A234">
  <a:p_name>HarpersCollins</a:p_name>
  <a:city>HarpersCollins</a:city>
</rdf:Description>
```

Internal = these resources are *not* visible outside

```
<http://.../isbn/2020386682> a:publisher _:A234.
_:A234 a:p_name "HarpersCollins".
```
RDF in programming practice

- For example, using Java+Jena (HP’s Bristol Lab):
  - a “Model” object is created
  - the RDF file is parsed and results stored in the Model
  - the Model offers methods to retrieve:
    - triples
    - (property,object) pairs for a specific subject
    - (subject,property) pairs for specific object
    - etc.
  - the rest is conventional programming...
- Similar tools exist in Python, PHP, etc.

---


- **RDF**
  - RDF API
  - Interact with the core API to create and read Resource Description Framework (RDF) graphs. Serialise your triples using popular formats such as RDF/XML or Turtle.
  - ARQ [SPARQL]
  - Query your RDF data using ARQ, a SPARQL 1.1 compliant engine. ARQ supports remote federated queries and free text search.
- **Triple store**
  - TDB
  - Persist your data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs.
  - Fuseki
  - Expose your triples as a SPARQL end-point accessible over HTTP. Fuseki provides REST-style interaction with your RDF data.
- **OWL**
  - Ontology API
  - Work with models, RDFS and the Web Ontology Language (OWL) to add extra semantics to your RDF data.
  - Inference API
  - Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS reasoners.
Jena example

```java
// create a model
Model model=new ModelMem();
Resource subject=model.createResource("URI_of_Subject")
// 'in' refers to the input file
model.read(new InputStreamReader(in));
StmtIterator iter=model.listStatements(subject,null,null);
while(iter.hasNext()) {
    st = iter.next();
    p = st.getProperty();
    o = st.getObject();
    do_something(p,o);
}
```

Another relatively simple application

- **Goal**: reuse of older experimental data
- **Keep data in databases** or XML, just export key “fact” as RDF
- **Use a faceted browser** to visualize and interact with the result

*Exhibit: [http://www.simile-widgets.org/exhibit/](http://www.simile-widgets.org/exhibit/)
One level higher up
(RDFS, Datatypes)

Need for RDF schemas

- First step towards the “extra knowledge”:
  - define the terms we can use
  - what restrictions apply
  - what extra relationships are there?
- Officially: “RDF Vocabulary Description Language”
  - the term “Schema” is retained for historical reasons...
RDF Schema (RDFS) - The RDF Vocabulary Description Language

- RDF Schema ‘semantically extends’ RDF to enable us to talk about classes of resources, and the properties that will be used with them.
- It does this by giving special meaning to certain rdf properties and resources.
- RDF Schema provides the means to describe application specific RDF vocabularies.

Describing Classes with RDFS

- To describe classes we can use built in RDF Schema resources:
  - rdfs:Class
  - rdfs:subClassOf
- These are used in conjunction with the rdf:type property.
Describing Properties with RDF(S)

- RDF Schema allows us to describe properties. (Properties are instances of the class `rdf:Property`!)
- We can specify a domain using `rdfs:domain`.
- We can specify a range using `rdfs:range`.

Other RDFS Built-In Properties

- `rdfs:subPropertyOf`
- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
Classes, resources, ...

- Think of well known traditional vocabularies:
  - use the term “novel”
  - “every novel is a fiction”
  - “«The Glass Palace» is a novel”
  - etc.
- RDFS defines resources and classes:
  - everything in RDF is a “resource”
  - “classes” are also resources, but...
  - ...they are also a collection of possible resources (i.e., “individuals”)
    - “fiction”, “novel”, ...

Classes, resources, ... (cont.)

- Relationships are defined among resources:
  - “typing”: an individual belongs to a specific class
    - “«The Glass Palace» is a novel”
    - to be more precise: “«http://.../000651409X» is a novel”
  - “subclassing”: all instances of one are also the instances of the other (“every novel is a fiction”)
- RDFS formalizes these notions in RDF
Classes, resources in RDF(S)

- RDFS defines the meaning of these terms
  - (these are all special URI-s, we just use the namespace abbreviation)

Inferred properties

- is not in the original RDF data...
- ...but can be inferred from the RDFS rules
- RDFS environments return that triple, too
Inference: let us be formal...

- The RDF Semantics document has a list of (33) entailment rules:
  - “if such and such triples are in the graph, add this and this”
  - do that recursively until the graph does not change
- The relevant rule for our example:

```xml
If:
  uuu rdfs:subClassOf xxx .
  vvv rdf:type uuu .
Then add:
  vvv rdf:type xxx .
```

Properties

- Property is a special class (rdf:Property)
- properties are also resources identified by URI-s
- There is also a possibility for a “sub-property”
  - all resources bound by the “sub” are also bound by the other
- Range and domain of properties can be specified
  - i.e., what type of resources serve as object and subject
Example for property characterization

=title
    rdf:type rdf:Property;
    rdfs:domain :Fiction;
    rdfs:range rdfs:Literal.

What does this mean?

- Again, new relations can be deduced. Indeed, if

  =title
    rdf:type rdf:Property;
    rdfs:domain :Fiction;
    rdfs:range rdfs:Literal.


- then the system can infer that:

  <http://.../isbn/000651409X> rdf:type :Fiction .
Literals

- Literals may have a data type
  - floats, integers, booleans, etc, defined in XML Schemas
  - full XML fragments
- (Natural) language can also be specified

Examples for datatypes

```
<http://.../isbn/000651409X> 
  :page_number "543"^^xsd:integer ;
  :publ_date   "2000"^^xsd:gYear ;
```
How to get and create RDF Data?

Simple approach

- Write RDF/XML, RDFa, or Turtle “manually”
- In some cases that is necessary, but it really does not scale...
RDF with XHTML

- Obviously, a huge source of information
- By adding some “meta” information, the same source can be reused for, eg, data integration, better mashups, etc
  - typical example: your personal information, like address, should be readable for humans and processable by machines

RDF with XML/(X)HTML (cont)

- Two solutions have emerged:
  - use microformats and convert the content into RDF
    - XSLT is the favorite approach
  - add RDF-like statements directly into XHTML via RDFa
Bridge to relational databases

- Data on the Web are mostly stored in databases
- “Bridges” are being defined:
  - a layer between RDF and the relational data
    - RDB tables are “mapped” to RDF graphs, possibly on the fly
    - different mapping approaches are being used
  - a number RDB systems offer this facility already (e.g., Oracle, OpenLink, ...)
- W3C has some work on this area:
  https://www.w3.org/2001/sw/rdb2rdf/

RDB2RDF Translation approaches and tools

- Check out this technical report:

  A survey of RDB to RDF translation approaches and tools. Franck Michel et al.