Referatsprotokolle

RDFPrimer

CC/PP

OWL Overview

Netzbasierte Informationssysteme

Elisa Barrotta
barrotta@inf.fu-berlin.de
10.02.2005

Table of Contents

1 RDFPrimer: Resource Description Framework Primer

Autors: Tobias Escher and Michail Starosta (20.01.2005)

1.1 RDF Introduction and Basic Concepts

RDF is one of the numerous W3C standards designed to offer a simpler representation of statements and data about resources. Resources can be described in terms of simple properties and values of these properties. RDF is not employed only for Websites.

Example in English: **http://www.example.org/index.html** has a **creator** whose value is **John Smith**. In this example the subject is the URL, the predicate is "creator" and the object is John Smith. However English cannot enable computers to communicate. We need a method for representing the subject, the predicate and the object and for exchanging information among computers.

To identify data RDF uses the URIs (Uniform Resource Identifiers), a more general form of URL (Uniform Resource Locator). URIs exist also for objects that do have not network locations (instead of URL).

Data can be represented by

1. a graph:
   - a node for the subject
   - a node for the object
   - an arc for the predicate, directed from the subject node to the object node;
2. a triple:
   - the subject, which can be an RDF URI reference or a blank node;
   - the predicate, which can be an RDF URI reference;
   - the object, which can be an RDF URI reference, a literal or a blank node. Example:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language>
"en" .
```

3. and a triple in XML Qname (Qualified Name). For example:

```
ex:index.html    dc:language        "en" .
```

In an RDF graph we can have a blank node. Blank nodes simplify the construction of complex relations and represent resources that do not have their own URIs through other URIs. When a blank node is used, in the triple there's a blank value that is resolved like _:language.

RDF doesn't define datatypes instead of programming languages. To identify literal types RDF codifies them with the URI. For example the age of a person:

```
exstaff:85740  exterms:age  "27"^^xsd:integer.
```

1.2 XML Syntax for RDF: RDF/XML

RDF provides an XML syntax for writing down graphs. For example:

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.              xmlns:dc="http://purl.org/dc/elements/1.1/"
4.              xmlns:exterms="http://www.example.org/terms/">
```

```
5.      <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
6.        <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
7.        <exterms:editor rdf:nodeID="abc"/>
8.      </rdf:Description>

9.      <rdf:Description rdf:nodeID="abc">
10.         <exterms:fullName>Dave Beckett</exterms:fullName>
11.         <exterms:homePage rdf:resource="http://purl.org/net/dajobe/"/>
12.     </rdf:Description>

13. </rdf:RDF>
```

Line 1 is the XML declaration. Line 2 with `rdf:RDF` indicates that the following XML content is intended to represent RDF and the XML namespace declarations at lines 3-4 specify that all tags in this content prefixed with `rdf:`, `dc:` and `exterms:` are part of the namespaces identified by the URIrefs . Line 5 indicates the start of a description of a resource and goes on to identify the resource the statement is about. Lines 6, 10 and 11 provide properties element. Line 7 indicates that there is a blank node to which we refer in RDF/XML using an `rdf:nodeID` attribute and we assign a blank node identifier `abc`. Line 12 indicates the end of this particular `rdf:Description` element and line 13 indicates the end of the rdf:RDF element.

1.3 RDF Capabilities

1.3.1 RDF Containers

For representing group of objects we can use the containers. There are three types of container:

- rdf:Bag, that denotes un unordered collection;
- rdf:Seq, that denotes un ordered collection;
- rdf:Alt, for alternative elements.

1.3.2 RDF Collections

An RDF collection is a group of things represented as a list structure in the RDF graph and describes a group containing only the specified objects.

1.3.3 RDF Reification

Reification allows us to treat an RDF-statement as the object of another RDF-statement.

1.4 RDF Schema

Vocabulary: numbers of concepts with a specific meaning (URIs).

Convention: the same prefix indicates belonging to the same theme.

Goal: create a common vocabulary for common understanding.

We need to refer to a vocabulary (through the URI) to create a common realm of understanding.

For example: Dublin Core (http://dublincore.org)

RDF Vocabulary Description Language 1.0: RDF provides a vocabulary to define

classes of objects and their properties. RDF Schema describes such classes and properties and indicates which classes and properties are expected to be used together.

A class is a group of things with common characteristics. Classes have specific properties defined through the element `rdf:Property`:

- `rdfs:domain` binds this properties in a fixed class;
- `rdfs:range` defines the values for this property ;
- `rdfs:subPropertyOf` specializes the property.

There are also RDF Schema Metainformations:

- `rdfs:comment`, adds comments;
- `rdfs:label`, adds names to the resource;
- `rdfs:seeAlso`, adds specific informations;
- `rdfs:isDefinedBy`: subproperty of seeAlso, defines the namespace of the subject.

RDF Schema provides also other capabilities for describing RDF vocabularies. These capabilities are target of the Ontology Languages.

1.5 RDF Applications

- Dublin Core Metadata Initiative
- PRISM
- RSS
- CIM/XML
- Gene Ontology
- CC/PP

2 CC/PP: Composite Capability/Preference Profile

Authors: Jussi Visapää and Viktoria Schwarzhaupt (27.01.2005)

2.1 Introduction

CC/PP is a standard profile language for representing the web contents with different devices and it is the first W3C recommendation that is an RDF application.

Goal: universal accessibility.

- It is independent from
  - Hardware
  - Software
  - Net Infrastructure
  - Language
  - Culture
  - Geographical position
- It increases the possibility of transmission of web contents between webservers and end devices.

CC/PP uses RDF taking advantage such:

- growing of vocabulary;
- decentralizing vocabulary;
- simplifying integration of data;
- placing the base of semantic web.

Specifications:

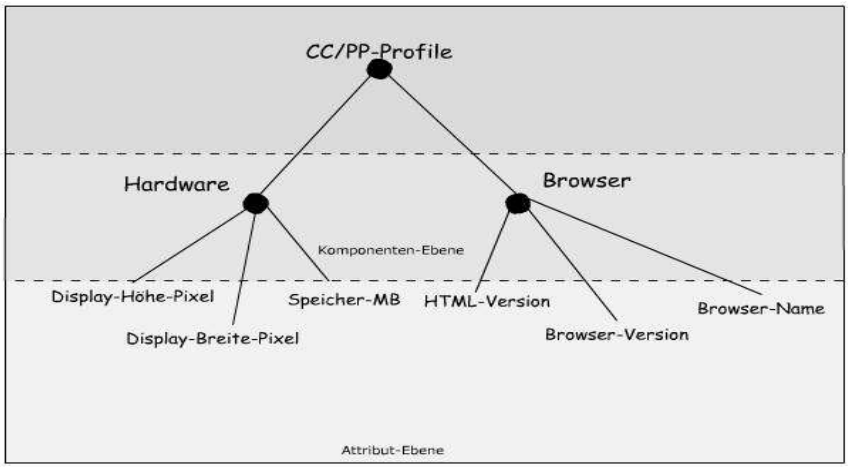
- A CC/PP profile is a description of devices properties and preferences.
- Clients are divided in hardware and software: memory, browser, supported html versions, internet protocols, plugins, JVM, operating system, etc.
- A server can use this information in a CC/PP profil for distributing to the clients a suitable representation of the requested datas.

2.2 Structure

A CC/PP profile is constructed as a two-level hierarchy:

- a profile having at least one or more components, for example software platform, browser, and
- each component having at least one or more attributes, for example version number, names, values.
- In addition: defaults that allows the use of default attributes.

CC/PP documents are represented through RDF/XML and the structure of this hierarchy is a graph:



2.2.1 Components

Each component is represented by a resource of type `ccpp:Component` or an RDF subclass and it may have an `rdf:type` property indicating what kind of client component it describes. If an attribute appears on different component types, the type MUST be indicated by an `rdf:type` property.

2.2.2 Attributes

The name of the attributes have to be valid XML, like

```
<ccpp:AttributeName>
Attribute value
</ccpp:AttributeName>
```

Attributes values can be simple or structured data types. For example:

- Simple types:
  - Strings: Case-sensitive (“Mozilla”, “5.0”,...)
  - Integer: Numbers belonging to $[-2^{31},+2^{31}-1]$ (+256 , -256, 256, ...)
  - Rational numbers: numbers represented like the quotient of integer numbers (14/11, -200/401, ...)
- Structured types:
  - A set of RDF attributes:

    Collection of unordered values: `rdf:Bag`

    Collection of ordered values: `rdf:Seq`

2.2.3 Defaults

Each component of a client profile may indicate a collection of default attribute values. This collection of default values can be defined

- inline (in the same CC/PP document), or
- extern (via a URI in the CC/PP profile).

If an attribute is also in the main part of the client profile, the default value will be changed with the given attribute value. Default values are referenced by the property `ccpp:defaults`.

Example of CC/PP document:

```
<?XML version=„1.0“?>
<rdf:RDF       ...>
<rdf:Description rdf:about="http://example.com/Profil">

<ccpp:component>
<rdf:Description rdf:about="http://example.com/TerminalHardware">
   <rdf:type rdf:resource="http://example.com/Schema#HardwarePlattform"/>
   <ccpp:defaults        rdf:resource="http://example.com/HardwareDefaults"/>
<ex:memoryMb>64</ex:memoryMb>
</rdf:Description>
</ccpp:component>

</rdf:Description>
</rdf:RDF>
```

2.3 Example of the use of CC/PP: User Agent Profile Specification (UAProf)

UAProf is a WAP Forum specification that is designed to allow wireless mobile devices to declare their capabilities to data servers and other network components. The design of UAProf is already based on RDF. Its vocabulary elements use the same basic format that is used for CC/PP.

The goal is that valid UAProf profiles are also valid CC/PP profiles; however not all CC/PP profiles are necessarily valid UAProf profiles.

2.4 Use of CC/PP in the future

CC/PP will be used increasingly in many mobile telephones due to the help of Java API for CC/PP. It has been announced that will be revised and integrated the RDF datatyping version.

3 OWL: Web Ontology Language Overview

Authors: Tatiana Sokolovskaya and Igor Kirianov (03.02.2005)

3.1 Introduction

Ontology: amount of ideas which describe a vocabulary for exchanging information. It is used to enlarge the existent web-based applications and to make possible a new kind of use of the web (web portal, collection of multimedia, cooperative management of websites, design of documentation).

The Web Ontology Language is another W3C standard used by applications when they need to process informations and not just to represent them in human understandable form. It is defined a complex ontology because

- it represents the meaning of terms in vocabularies and the relationships between those terms, and
- it uses the XML, XML Schema, RDF and RDF Schema rules adding more vocabularies for describing properties and classes.

With OWL, and the vocabulary of RDF, we can express that

- classes are disjointed;
- classes are defined like boolean combinations of other classes;
- properties have restriction of cardinality;
- properties can be transitive, simmetric or functional.

Unfortunately OWL is not at all compatible with RDF.

3.2 Sublanguages of OWL

OWL has three sublanguages that are each the extension of its simpler predecessor. The use of one instead of the other is due to the needs of the developpers.

3.2.1 OWL Lite

OWL Lite is used to have hierarchy classification and simple constraints. It also have a lower formal complexity than the others. Properties:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.

The inverse is not allowed.

3.2.2 OWL DL

OWL DL comes from *description logics,* a field of research that has studied the logics that form the formal foundation of OWL. It is used by users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). Properties:

- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

The inverse is not not possible.

3.2.3 OWL Full

OWL Full support those users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

3.3 Language description of OWL Lite

Description of OWL features.

3.3.1 RDF Schema Features

- *Class*: defines a group of individual with the same properties, for example: <owl:Class rdf:ID=„person“>;
- *rdfs:subClassOf*: to create hierarchies of classes, for example <rdfs:subClassOf rdf:Resource=„#nature“>;
- *rdf:Property*: properties used to state the relationships between individuals and individuals and values;
- *rdfs:subPropertyOf*: to create hierarchies of properties;
- *rdfs:domain*: to define which properties have to be applied to an individual, for example <rdfs:domain rdf:Resource=„#person“>;
- *rdfs:range*: limits the individuals that the property may have as its value, for example <rdfs:range rdf:Resource=„xsd:decimal“>;
- *Individual*: instances of classes, <owl:Class rdf:ID=„person“> <Person rdf:ID= „Klaus“>.

3.3.2 Equality and Inequality

- *equivalentClass*: two classes are equivalent;
- *equivalentProperty*: two properties are equivalent;
- *sameAs*: two individuals are the same;
- *differentFrom*: two individuals are different;
- *allDifferent*: all the individuals are mutually distinct.

3.3.3 Property Characteristics

- *InverseOf*: one property can be defined as the inverse of another;
- *TransitiveProperty*: define a property as transitive;
- *SymmetricProperty*: define a property as symmetric;
- *FunctionalProperty*: a property can have a unique value;
- *InverseFunctionalProperty*: a property can be defined as the inverse of another.

3.3.4 Property Restrictions

- *AllValuesFrom*: refers only to a class that has a local range restriction associated with the property;
- *someValuesFrom*: refers only to a class in which at least one value for that property is of a certain type.

3.3.5 Restricted Cardinality

OWL Lite allows only cardinalities of values 0 and 1.

- *minCardinality*: stated on a property with respect to a particular class;
- *MaxCardinality*: stated on a property with respect to a particular class;
- *cardinality*: useful when that property on a class has both *minCardinality* 0 and *maxCardinality* 0 or both *minCardinality* 1 and *maxCardinality* 1.

3.3.6 Class Intersection

- IntersectionOf: OWL Lite allows intersections of named classes and restrictions.

3.4 Addition in OWL DL and OWL Full

OWL DL and OWL Full adds other features:

- *oneOf*: enumeratates the individuals that make up the class.
- *hasValue*: properties can be required to have a certain individual as a value.
- DisjointWith: classes can be disjoint from each other.
- UnionOf, complementOf, intersectionOf: arbitrary Boolean, combinations of classes and restrictions can be allowed.
- MinCardinality, maxCardinality, cardinality: arbitrary values for cardinality can be alloed.
- complex classes: OWL Full allows arbitrarily complex class descriptions consisting of enumerated classes, property restrictions and Boolean combinations, while OWL Lite restricts the syntax to single class names.