

IN3060/4060 – Semantic Technologies – Spring 2021

Lecture 2: Resource Description Framework (RDF)

Jieying Chen

22nd January 2021



DEPARTMENT OF
INFORMATICS



UNIVERSITY OF
OSLO

Today's Plan

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties
- 7 Summary

Mandatory exercises

- First oblig published today (22.01) after lecture.
- Topic RDF.
- Hand in by next Friday (29.01).
- Same schedule for the other small obligs:
 - #2 (29.01 – 05.02),
 - #3 (05.02 – 12.02), and
 - #4 (19.02 – 05.03).
- The larger obligs with two possible attempts:
 - #5 (05.03 – 19.03) and
 - #6 (26.03 – 16.04).
- And one short oblig about OTTR
 - #7 (07.05 – 14.05).
- See *obliger* on the semester page.
- Mr. Oblig.

Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties
- 7 Summary

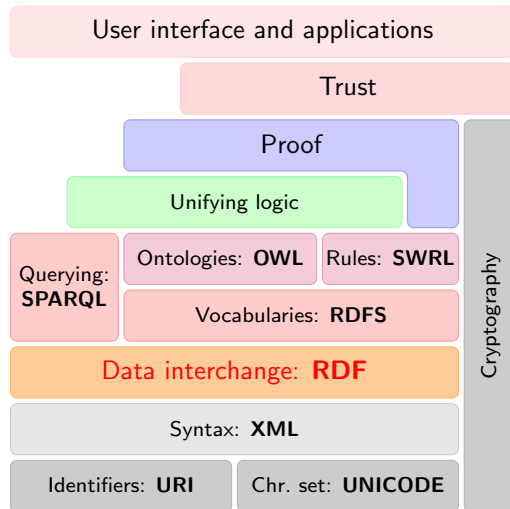
RDF: W3C Overview

- *The Resource Description Framework* (RDF) is a standard model for data interchange on the Web.
- It has features that facilitate data merging even if the underlying schemas differ.
- It extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link.
- Thus allows data to be mixed, exposed, and shared across different applications.
- This linking structure forms a directed, labelled graph.
- This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.

Adapted from <http://w3c.org/RDF>.

Semantic Web Stack

- Central block in the SW stack.
- First “semantic” block in stack.
- In the course we will explore:
 - RDF
 - SPARQL
 - RDFS/OWL
 - Logic
 - Applications



RDF, essential 'abouts':

- The *Resource Description Framework* was initially intended for annotation of web-accessible resources (1999).
- It has since developed into a general purpose language for describing structured information—on the web or elsewhere.
- The goal of RDF is to enable applications to exchange data on the Web in a meaning-preserving way.
- It is considered the basic representation format underlying the Semantic Web.

Outline

- 1 Introduction
- 2 **RDF data model**
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties
- 7 Summary

RDF Triples

- All information in RDF is expressed using a *triple* pattern.
- A triple consists of a **subject**, a **predicate**, and an **object**.

Examples:

subject	predicate	object
Norway	has capital	Oslo
Norway	has king	King Harald
King Harald	born year	1937

- Another word for an RDF triple is a *statement* or *fact*.
- The elements of an RDF triple are either
 - *URI references*,
 - *literals*, or
 - *blank nodes*.

Uniform Resource Identifiers (URIs)

- RDF (Resource Description Framework) talks about *resources*.
 - Almost anything is a resource.
- Resources are identified by URIs (Uniform Resource Identifiers).
- E.g., in `dbpedia.org`:
 - Norway: `http://dbpedia.org/resource/Norway`
 - has capital: `http://dbpedia.org/ontology/capital`
 - Oslo: `http://dbpedia.org/resource/Oslo`
 - has king: `http://dbpedia.org/ontology/leader`
 - King Harald: `http://dbpedia.org/resource/Harald_V_of_Norway`
- As identifiers, think of them as just strings (on a special format).
 - Not necessarily dereferenceable.

URI $\not\subseteq$ URL

URLs are not the only URIs:

- ISBN:
`urn:isbn:0-486-27557-4`
- Geo:
`geo:37.786971,-122.399677`
- Mail:
`mailto:jieyingc@ifi.uio.no`
- and many many more ...

URIs and QNames

- URIs are often long and hard to read and write.
- Most serialisations use an abbreviation mechanism.
 - Define “prefixes”, “namespaces”.
- E.g., in Turtle serialisation:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix dbp-ont: <http://dbpedia.org/ontology/> .
```
- A *QName* like `dbp:Oslo` stands for `http://dbpedia.org/resource/Oslo`
- Remember: It's all just URIs!

URIs and data

- We can then state that Norway's capital is Oslo as:

```
<http://dbpedia.org/resource/Norway> <http://dbpedia.org/ontology/capital> <http://dbpedia.org/resource/Oslo> .
```

- Or use prefixes:

```
dbp:Norway dbp-ont:capital dbp:Oslo .
```

- But what if we want to state that Oslo's population is 629313?
- We cannot have one URI for every integer, decimal number, string etc.

Literals

- Literals are used to represent data values.
- All literals have a datatype.
- Datatypes are also resources, referenced via URIs, and written as:
`dbp:Oslo dbp-ont:population "629313"^^xsd:integer .`
- However, if nothing is written, it is assumed to be a string:
`dbp:Oslo dbp-ont:officialName "Oslo" .`
 Is short for
`dbp:Oslo dbp-ont:officialName "Oslo"^^xsd:string .`
- One can also specify the language of a string using a *language tag*:
`dbp:Norway rdfs:label "Norge"@no .`
`dbp:Norway rdfs:label "Norwegen"@de .`

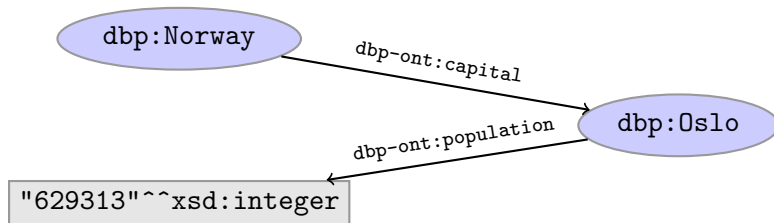
RDF Graphs

- An *RDF graph* is a set of triples. E.g.,

```
dbp:Norway dbp-ont:capital dbp:Oslo .
dbp:Oslo dbp-ont:population "629313"^^xsd:integer .
```

is an RDF graph containing two triples.

- RDF graphs are often represented as a directed labelled graph:



Problems

- Can all knowledge be nicely represented with only triples containing URIs and literals?
- What if we didn't know what the capital of Norway was, only that it has a population of 629313 people?

```
dbp:Norway dbp-ont:hasCapitalWithPopulation "629313"^^xsd:integer .
```

- What if we want to model something which is not nicely represented as one URI, e.g. an address?
- UiO has the address "Problemveien 7 0313 Oslo". How should we model this? As a literal?

```
dbp:UiO dbp-ont:hasAddress "Problemveien 7 0313 Oslo" .
```

- As several literals?

```
dbp:UiO dbp-ont:addressPlace "Oslo" .
```

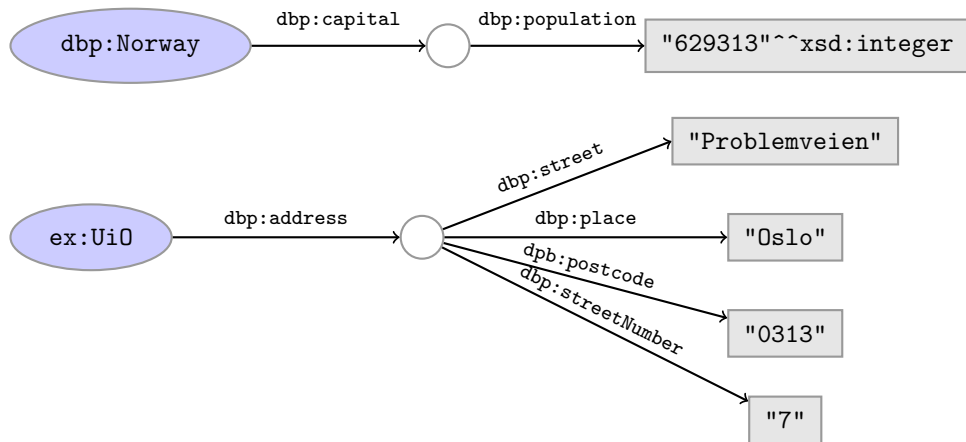
```
dbp:UiO dbp-ont:addressStreet "Problemveien" .
```

```
dbp:UiO dbp-ont:addressStreetNumber "7" .
```

```
dbp:UiO dbp-ont:addressPostcode "0313" .
```


Blank nodes

- Blank nodes are like resources without a URI.
- Use when resource is unknown, or has no (natural) identifier. E.g.:



RDF Triple Grammar

- Literals and blank nodes may not appear everywhere in triples:

	s	p	o
● URI references may occur in all positions	✓	✓	✓
● Literals may only occur in object position	✗	✗	✓
● Blank nodes may not occur in predicate position	✓	✗	✓

- Why?

- Literals are just values, no relationships from literals allowed.
- Blank nodes in predicate position deemed “too meaningless” and confusing.

Why URIs?

- URIs naturally have a “global” scope, unique throughout the web.
 - Contrasts to, e.g., keys in rel. DB which are unique within a table.
 - Helps to avoid name clashes.
 - Example: merging two product catalogues.
`http://www.abc-company.com/category/item/123`
`http://www.xyz-company.com/product/123`
- URLs are also addresses.
 - Exploit the well-functioning machinery of web browsing.
 - Find data by following data identifiers, i.e., URIs.
- *“A web of data.”*

Why triples?

- Any information format can be transformed to triples.
 - Examples:
 - Tabular (spreadsheets, DBs): **row** **column** **cell**
 - Trees (XML): **parent** **path** **child**
- Relationships are made explicit and elements in their own right.
 - The predicate, i.e., the relationship, is an element in the triple.
 - Unlike DB columns and binary predicates.
 - Can be described in RDF.
 - “Self-documenting”.
- Again, “*A web of data*”.

Why graphs?

- A single, but highly versatile, format.
 - Everything is on the same format: triples!
- Since RDF graphs are just sets of triples, basic set operations are well-defined.
- Merging RDF graphs? Just take their union!
 - With tabular data, table dimensions must match.
 - With trees, a node can only have one parent.
 - Note that graphs need not be connected.
- Extending an RDF graph? Just add more triples!
 - Need not redefine the database table, or
 - to restructure the XML schema.

Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations**
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties
- 7 Summary

RDF Serialisations

There are many serialisations for the RDF data model:

RDF/XML the W3C standard. Complicated!

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Harald_V_of_Norway">
    <foaf:name>Harald V</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

Turtle convenient, human readable/writable—our choice.

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

dbp:Harald_V_of_Norway foaf:name "Harald V" .
```

N-triples one triple per line. No abbreviations.

```
<http://dbpedia.org/resource/Harald_V_of_Norway> <http://xmlns.com/foaf/0.1/name> "Harald V" .
```

Others N3, TriX, TriG, RDF/JSON, ...

Turtle: URI references and triples

Full URIs are surrounded by `<` and `>`:

```
<http://dbpedia.org/resource/0slo>
```

Statements are triples terminated by a period:

```
<http://dbpedia.org/resource/0slo>  
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
  <http://dbpedia.org/ontology/Place> .
```

Use 'a' to abbreviate `rdf:type`:

```
<http://dbpedia.org/resource/0slo>  
  a <http://dbpedia.org/ontology/Place> .
```

Turtle allows any non-zero amount of space between elements in triples.

Turtle: Namespaces

QNames are written without any special characters.

Namespace prefixes are declared with @prefix:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
  
dbp:Oslo a <http://dbpedia.org/ontology/Place> .
```

A default namespace may be declared:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix : <http://dbpedia.org/ontology/> .  
  
dbp:Oslo a :Place .
```

Turtle: Literals

Literal values are enclosed in double quotes:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix : <http://dbpedia.org/ontology/> .  
  
dbp:Norway :officialName "Norge" .
```

Possibly with type or language information:

```
dbp:Norway rdfs:label "Norge"@no .  
dbp:Oslo :population "629313"^^xsd:integer .
```

Numbers and booleans may be written without quotes:

```
dbp:Oslo :population 629313 .  
dbp:Oslo :isCapital true .
```

Turtle: Statements sharing elements

Instead of:

```
dbp:Oslo rdf:type dbo:City .  
dbp:Oslo :officialName "Oslo" .  
dbp:Oslo :population 629313 .
```

...statements may share a subject with ';':

```
dbp:Oslo rdf:type dbo:City ;  
         :officialName "Oslo" ;  
         :population 629313 .
```

Turtle: Statements sharing elements

Instead of:

```
dbp:Norway rdfs:label "Norway"@en .  
dbp:Norway rdfs:label "Norwegen"@de .  
dbp:Norway rdfs:label "Norge"@no .
```

... statements may share subject and predicate with ',':

```
dbp:Norway rdfs:label "Norway"@en ,  
                "Norwegen"@de ,  
                "Norge"@no .
```

... and in combination:

```
dbp:Norway rdfs:label "Norway"@en, "Norwegen"@de, "Norge"@no ;  
           :capital dbp:Oslo .
```

Turtle: Blank nodes

Blank nodes are designated with underscores or [...].

Norway has a capital with population 629313:

```
dbp:Norway :capital _:someplace .  
_:someplace :population 629313 .
```

There is a place with official name Oslo:

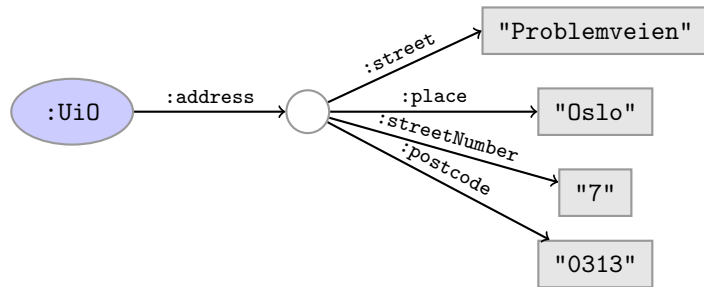
```
[] a :Place ;  
   :officialName "Oslo" .
```

UiO has address Problemveien 7, 0313 Oslo:

```
:UiO :address [ :street "Problemveien" ;  
                :streetNumber "7";  
                :place "Oslo" ;  
                :postcode "0313" ] .
```

Question

The blank node here:



has no 'name.'

Why does Turtle use 'blank node identifiers' like `_:someplace`?

Answer: makes it easy to use same node in several triples.

Turtle: Other things

Use '#' to comment:

```
# This is a comment.  
dbp:Oslo a dbpont:Place . # This is another comment.
```

Use '\' to escape special characters:

```
:someGuy :foaf:name "James \"Mr. Man\" Olson" .
```

Turtle specification: <http://www.w3.org/TR/turtle/>.

Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies**
- 5 RDF on the web
- 6 Subtleties
- 7 Summary

Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Usually the same namespace/prefix is shared.
- Some important, well-known namespaces—and prefixes:

rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> – RDF

rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> – RDF Schema

foaf: <<http://xmlns.com/foaf/0.1/>> – Friend of a friend

dcterms: <<http://purl.org/dc/terms/>> – Dublin Core

- Usually, a description is published at the namespace base URI.
- Note that the prefix is not standardised.
 - However, in practice many are.
 - @prefix rdf: <<http://xmlns.com/foaf/0.1/>> would be highly irregular.

Example vocabularies: RDF, RDFS

Some example resources:

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`,
`rdf:predicate`,
`rdf:object`
- `rdf:type`

RDFS: describing RDF vocabularies.

- `rdfs:Class`
- `rdfs:subClassOf`,
`rdfs:subPropertyOf`
- `rdfs:domain`,
`rdfs:range`
- `rdfs:label`

Examples:

```
dbp:Oslo rdf:type dbp-ont:Place .  
dbp:Norway rdfs:label "Norge"@no .  
dbp:Capital rdfs:subClassOf dbp:City .
```

Example vocabularies: FOAF, Dublin Core

Some example resources:

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`,
`foaf:lastName`,
`foaf:gender`

Dublin Core: library metadata.

- `dcterms:creator`,
`dcterms:contributor`
- `dcterms:format`,
`dcterms:language`,
`dcterms:licence`

Examples:

```
ifi:jieyingc rdf:type foaf:Person .
```

```
ifi:jieyingc foaf:knows ifi:martingi .
```

```
ifi:jieyingc dcterms:creator ifi:rdf-lecture .
```

Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web**
- 6 Subtleties
- 7 Summary

Where is it?

- In files:
 - In some serialisation: XML/RDF, Turtle, ...
 - Typically small RDF graphs, i.e., max. a few 100 triples, e.g.,
 - Vocabularies: <http://xmlns.com/foaf/spec/index.rdf>.
 - Tiny datasets: <http://folk.uio.no/martingi/foaf.rdf>.
- From *SPARQL endpoints*:
 - Data kept in a *triple store*, i.e., a database.
 - RDF is served from endpoint as results of *SPARQL queries*.
 - Exposes data (in different formats)
 - with endpoint frontends, e.g., <http://dbpedia.org/resource/Norway>, or
 - by direct SPARQL query: <http://dbpedia.org/sparql>.
- There are many *RDFizers* which convert data to RDF.
 - Tabular files (CSV, Excel): XLWrap.
 - Relational DB: D2RQ (<http://sws.ifi.uio.no/d2rq/>) or R2RML (<https://www.w3.org/TR/r2rml/>).
 - W3C keeps a list: <http://www.w3.org/wiki/ConverterToRdf>.

Creating RDF data and vocabularies

- Designing an easy-to-use and robust namespace is non-trivial.
- Naming is difficult.
- Reuse existing vocabularies if possible. Don't reinvent.
- URIs are also addresses, consider publishing issues when naming.
- Adhere to the policies described in *best practice* documents:
 - Best Practice Recipes for Publishing RDF Vocabularies
 - <http://www.w3.org/TR/2008/NOTE-swbp-vocab-pub-20080828/>
 - Cool URIs for the Semantic Web
 - <http://www.w3.org/TR/cooluris/>
- Use `http://www.example.[com|net|org]` for prototyping and documentation.

Linked Open Data

Tim Berners-Lee's recipe for 5 star web data:

- ① Make data available on the Web (any format) under an open license.
- ② Make it available as structured data (e.g., Excel, not image scans).
- ③ Use non-proprietary formats (e.g., CSV instead of Excel).
- ④ Use URIs to identify data items; make them referable on the Web.
- ⑤ Link your data to other's data to provide context.

Adapted from <http://www.w3.org/DesignIssues/LinkedData.html>.

Web of Data

- The point of publishing data as described in this lecture is to have self-describing and self-documenting data.
- Decouples data from applications.
- Lightens the programming burden.
- Semantic Web applications should be/are generic and general purpose, exploiting rich and knowledge intensive data sets.

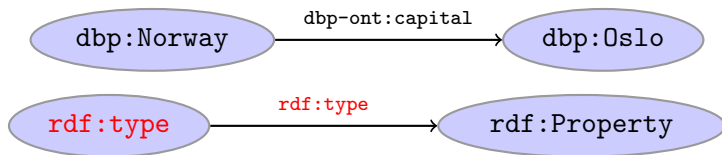
Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties**
- 7 Summary

URIs are not necessarily unique

- URIs are just strings, not a “global identification service”.
- There is nothing stopping you from using `rdf:type` as the URI for your favourite data item.
- However, don't do that!
- The simple rule of only creating URIs in a namespace domain you control should keep you out of trouble.
 - Again, put data on the URI address.
- *Trust* is an important (and work-in-progress) layer in the SW stack.

RDF graphs are not graphs



- Drawing `dbp:Norway dbp-ont:capital dbp:Oslo` is straight-forward.
- But what about `rdf:type rdf:type rdf:Property`?
- RDF graphs are sets of triples, not graphs.
- The set of nodes, i.e., subjects and object, and edges, i.e., predicates, of an RDF graph need not be disjoint.

Be careful when merging RDF *files*

Merging the two RDF files containing named blank nodes

File 1

```
ifi:martige :owns _:myCar .
_:myCar a lotus:Esprit .
```

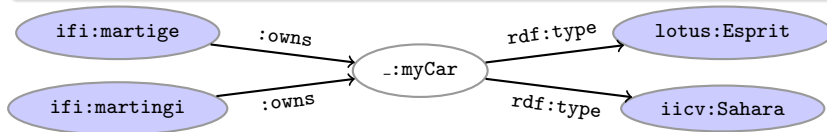
File 2

```
ifi:martingi :owns _:myCar .
_:myCar a iicv:Sahara .
```

gives the RDF graph:

File 1 \cup File 2

```
ifi:martige :owns _:myCar .
ifi:martingi :owns _:myCar .
_:myCar a lotus:Esprit, iicv:Sahara .
```



Rename blank nodes

Renaming `_:myCar` to `_:myCar2` in File 2.

File 1

```
ifi:martige :owns _:myCar .
_:myCar a lotus:Esprit .
```

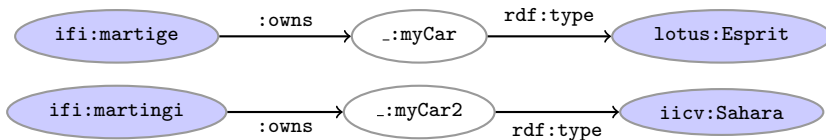
File 2

```
ifi:martingi :owns _:myCar2 .
_:myCar2 a iicv:Sahara .
```

gives the RDF graph:

File 1 \cup File 2

```
ifi:martige :owns _:myCar . _:myCar a lotus:Esprit .
ifi:martingi :owns _:myCar2 . _:myCar2 a iicv:Sahara .
```



More complex statements

We can use triples to form complex statements, e.g.:

Data structures

```

:in3060 :hasLecturers
  [ rdf:first :martingi ;
    rdf:rest [ rdf:first :jieyingc;
               rdf:rest [ rdf:first :olemholt ;
                           rdf:rest rdf:nil .
                         ] .
          ] .
  ] .

```

Turtle shorthand for lists

```

:in3060 :hasLecturers (:martingi :jieyingc :olemholt ) .

```

More complex statements (cont.)

What if I want to state that “Jieying thinks iOS is better than Linux, but Martin does not.”

Reification, statements describing statements

```
_:s rdf:subject ex:ios ;  
    rdf:predicate ex:betterThan ;  
    rdf:object ex:linux .  
  
:jieyingc :thinks _:s .  
:martingi :thinksNot _:s .
```

Reification allows us to describe agents' (e.g. people, sensors) beliefs, knowledge, etc. or meta information about a statement, e.g. “added by”, “imESTAMP”, etc.

Outline

- 1 Introduction
- 2 RDF data model
 - Technicalities
 - Features
- 3 RDF serialisations
- 4 RDF vocabularies
- 5 RDF on the web
- 6 Subtleties
- 7 Summary**

Summary

- RDF is a general format for describing resources.
- Data is represented as triples, consisting of
 - URIs for describing resources,
 - literals for data,
 - blank nodes for unknown data or more complex relationships.
- Sets of triples form RDF graphs.
- Naturally extends the linking structure of the web.
- Allows meta-data as a part of the data.
- Allows data to be easily linked to other datasets.
- Is completely independent of any application.

That's it for today!

Remember the mandatory assignment.