

INFO216: Advanced Modelling

Theme, spring 2017:
Modelling and Programming
the Web of Data

Andreas L. Opdahl
<Andreas.Opdahl@uib.no>



Session 4: Application architecture

- Themes:
 - application architecture for the web of data
 - components of web-of-data applications
 - programming against TDB in Jena
 - basic OWL concepts (“RDFS Plus”)



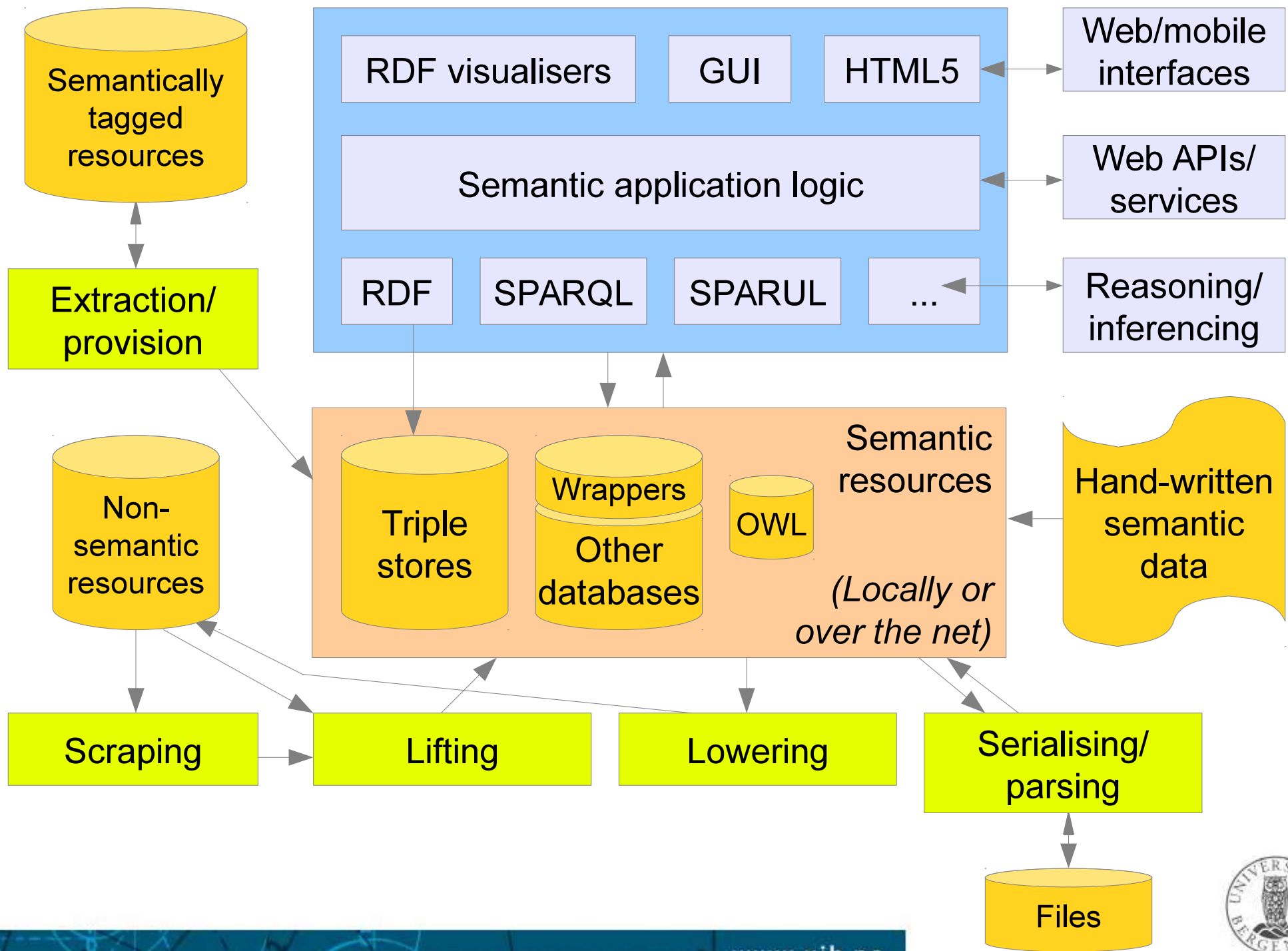
Readings

- Sources:
 - Allemang & Hendler (2011):
Semantic Web for the Working Ontologist,
 - chapter 4 on application architecture
 - chapter 8 on RDFS Plus
 - materials at wiki.uib.no/info216



Application architecture for the Web of Data





Parsing/serialising

- Reading from (“parsing”) and writing to (“serialising”) standard RDF formats
- Why different formats?
 - compactness, XML-dependency
 - can the same data set be stored in many ways?
 - machine versus human readability, abbreviations
 - CURIEs (“Compact URIs”) with qname/prefix
 - nested resources
 - scope: only basic RDF or also, e.g., quads, rules , OWL...
- Built into all RDF- (and OWL-) programming frameworks
 - e.g. Jena



Example: TURTLE

- *“Terse RDF Triple Language”*
 - extends the N-Triple format
 - restricts the Notation 3 (N3) -format
 - not XML-based (like RDF/XML), but simpler to read
 - *supports prefixes* (and bases)
 - *writing multiple predicates-objects for the same subject*
 - *writing multiple objects for the same subject-predicate*
 - flexible notations for blank/anonymous nodes: `[]`, `[...]`
 - TriG extends TURTLE to support named graphs/quads
 - SPARQL uses TURTLE-like syntax
 - OWL is sometimes written in TURTLE
 - *but OWL also has its own notations!*



Example: TURTLE

N-TRIPLE:

```
<http://r.e.x/Harald> <http://r.e.x/ektefelle> <http://r.e.x/Sonja> .  
<http://r.e.x/Harald> <http://r.e.x/barn> <http://r.e.x/Haakon_Magnus> .  
<http://r.e.x/Harald> <http://r.e.x/barn> <http://r.e.x/Martha_Louise> .
```

TURTLE:

```
<http://r.e.x/Harald> <http://r.e.x/ektefelle> <http://r.e.x/Sonja> ;  
    <http://r.e.x/barn> <http://r.e.x/Haakon_Magnus> ,  
    <http://r.e.x/Martha_Louise> .
```

- *semicolon (;) means “new predicate, same subject”*
- *comma (,) means “new object, same subject, predicate”*
- *period (.) means “new subject”*



Example: TURTLE

TURTLE:

```
@prefix rex: <http://r.e.x/> .
```

```
<rex:Harald> <rex:ektefelle> <rex:Sonja> ;  
    <rex:barn> <rex:Haakon_Magnus> ,  
    <rex:Martha_Louise> .
```

- *@prefix allows use of Compact URIs (“Curies”)*
- *@base allows use of IRI-fragments*
- *we will look at blank/anonymous nodes later...*



Example: TriG

TriG:

```
@prefix rex: <http://r.e.x/> .
```

```
<rex:Royal> { <rex:Harald> <rex:spouse> <rex:Sonja> ;  
              <rex:kid> <rex:Haakon_Magnus> ,  
                  <rex:Martha_Louise> . }
```

```
<rex:Mine> { <reg:Andreas> <reg:spouse> <reg:Margareth> ;  
            <reg:kid> <reg:Jens_Christian> . }
```

- *extends Turtle with named graphs wrapped in { ... }*



Important RDF serialisations

- RDF/XML (*the original XML serialisation*)
- TriX (*XML-based, experimental, named graphs*)
- N-TRIPLE (*maximally simple format, has “canonical form”*)
- NQ, NQUAD (*extends N-TRIPLE with quads*)
- TURTLE (“Terse RDF Triple Language”)
(*builds on N-TRIPLE, human readable, SPARQL ++*)
- TriG (*TURTLE-extension, named graphs*)
- Notation3, N3 (*builds on TURTLE, supports rules, graphs ++*)
- JSON-LD (*“JavaScript Object Notation – Linked Data”*)
- embedded formats:
 - microformats, (eRDF →) RDFa, microdata
- In addition, OWL has its own serialisations...
 - RDF/XML and TURTLE are sometimes used



Example: JSON

```
{  
  "homepage": "http://me.markus-lanthaler.com", ← Member  
  "name": "Markus Lanthaler",  
  "workplaceHomepage": "http://www.tugraz.at/"  
}
```

Object Name Value

JavaScript Object Notation (JSON)
www.json.org



Example: JSON

```
{  
  "homepage": "http://me.markus-lanthaler.com",  
  "name": "Markus Lanthaler",  
  "workplaceHomepage": "http://www.tugraz.at/"  
}
```

This is the person's id!

<http://xmlns.com/foaf/0.1/name>

<http://xmlns.com/foaf/0.1/workplaceHomepage>

<http://xmlns.com/foaf/0.1/Person>

How to represent semantic data in JSON?



Example: JSON-LD

```
{
  "homepage": "http://me.markus-lanthaler.com",
  "name": "Markus Lanthaler",
  "workplaceHomepage": "http://www.tugraz.at/"
}
```

This is the person's id!

<http://xmlns.com/foaf/0.1/name>

<http://xmlns.com/foaf/0.1/workplaceHomepage>

<http://xmlns.com/foaf/0.1/Person>

```
{
  "@type" : "http://xmlns.com/foaf/0.1/Person",
  "@id": "http://me.markus-lanthaler.com",
  "http://xmlns.com/foaf/0.1/name": "Markus Lanthaler",
  "http://xmlns.com/foaf/0.1/workplaceHomepage":
    { "@id" : "http://www.tugraz.at/" }
}
```

www.json-ld.org
Topic of lecture S05...



Scraping

- Making less structured data locally available in a well-structured format
- Typically used on internet data:
 - from less to more explicitly structured formats
 - HTML, PDF, DOCX, TXT, tagged file formats
- Storing the result in, e.g., CSV, XML or JSON
- A useful “technical craft”
 - not our focus
 - using scripts, regular expressions
 - *check what others have done before (jsoup)!*
 - *think continuous process – not once-off conversion!*



Semantic lifting

- Making structured data semantic
 - ...important for us
- Often the next step after scraping
 - ...or in parallel with scraping
 - storing the result in, e.g., RDF, RDFS, OWL...
- Tasks:
 1. **creating triples** (*make everything (s, p, o)-triples*)
 2. **creating graphs** (*one or several?*)
 3. **selecting IRIs** (*standard IRIs as identifiers*)
 4. **selecting vocabularies** (*standard IRIs as predicates*)
 5. **selecting types** (*standard IRIs as resource types*)
 6. **external linking** (*owl:sameAs*)



Extraction

- Retrieving RDF triples from (semantically) tagged resources
 - e.g., microformats, (eRDF ->) RDFa, microdata
- Replaces scraping + lifting
 - but is much simpler
 - the tags already do much of the job
 - open-source code is often available



Triple stores

- Basic software for persistent triple stores
 - or: database management systems (DBMSs) for RDF triples
 - general DBMS properties and behaviours
- Examples:
 - *Apache Jena TDB* (simple, file based, RDF-centric)
 - *Eclipse RDF4J (Sesame)* (much used, RDF-centric)
 - *Ontotext GraphDB (OWLIM)* (RDF4J compatible)
 - *Stardog* (RDF4J compatible)
 - *OpenLink Virtuoso* (much used, supports multiple data models, large)

<https://www.w3.org/wiki/LargeTripleStores>



Why different triple stores?

- A few important properties:
 - capacity (a trillion triples (norsk: “billion”, 10^{12}))
 - performance, security
 - SPARQL version (1.0, 1.1, Update)
 - SQL dependency, supports other data models?
 - FLOSS, license, price
 - in memory / on file
 - local server or cloud-hosted
 - single- / multi-thread and -server
 - reasoning
 - programming language
 - built-in SPARQL or other endpoints?



Visualisation

- APIs:
 - general GUI APIs
 - graph drawing/editing APIs
- Cloud based:
 - graph and general visualisers
 - e.g., embedded in web pages
 - often SPARQL-based
 - a SPARQL query extracts the dataset
 - the SELECTed variables are used to draw
 - graphs, bar charts, pie charts...
 - e.g., *<http://mgskjaeveland.github.io/sgvizler/>*



Endpoints

- Providing access to semantic resources over the net using standard protocols
 - typically HTTP, SPARQL, RDF, XML, JSON
 - based on
 - pure RDF resources, or
 - “wrapped” resources, e.g., relational databases
- Also simple web interfaces for interactive use
 - e.g., SNORQL (<http://dbpedia.org/snorql/>), Fuseki



Wrappers

- Wrapping existing structured data resources to present them as semantic resources
 - often relational data
 - but also, e.g., spreadsheets, XML, JSON
 - on-demand (live) semantic lifting
 - attributes/columns are mapped to predicates
 - read-only or read+update?
 - handwritten or wrapper software
 - e.g., D2RQ (<http://d2rq.org>)
 - wrapped resources can be used locally
 - or made accessible through an endpoint



Three-level architecture

- Raw data sets:
 - available in a standard format
 - perhaps virtually
 - SPARQL end points, RDF files
- Abstract data representation (RDF):
 - graph of nodes and arrows
- Queries:
 - standard query languages
 - based on the abstract data representation
- *Enabled by the semantic technologies*

