

# Programming

## THE SEMANTIC WEB

Build an application upon Semantic Web models. Brief overview of Apache Jena and OWL-API.



POLITECNICO  
DI TORINO



# Recap: Tools

- Editors (<http://semanticweb.org/wiki/Editors>)
  - Most common editor: **Protégé 5**
  - Other tools: TopBraid Composer (\$), NeOn toolkit
  - Special purpose apps, esp. for light-weight ontologies (e.g., FOAF editors)
- Reasoners (<http://semanticweb.org/wiki/Reasoners>)
  - OWL DL: Pellet 2.0\*, **HermiT**, FaCT++, RacerPro (\$)
  - OWL EL: CEL, SHER, snorocket (\$), ELLY
  - OWL RL: OWLIM, Jena, Oracle OWL Reasoner (\$)
  - OWL QL: Oowlgres, QuOnto, Quill

\* The next-gen reasoner (version 3) is part of Stardog, a closed source RDF database

# Recap: How to create an ontology



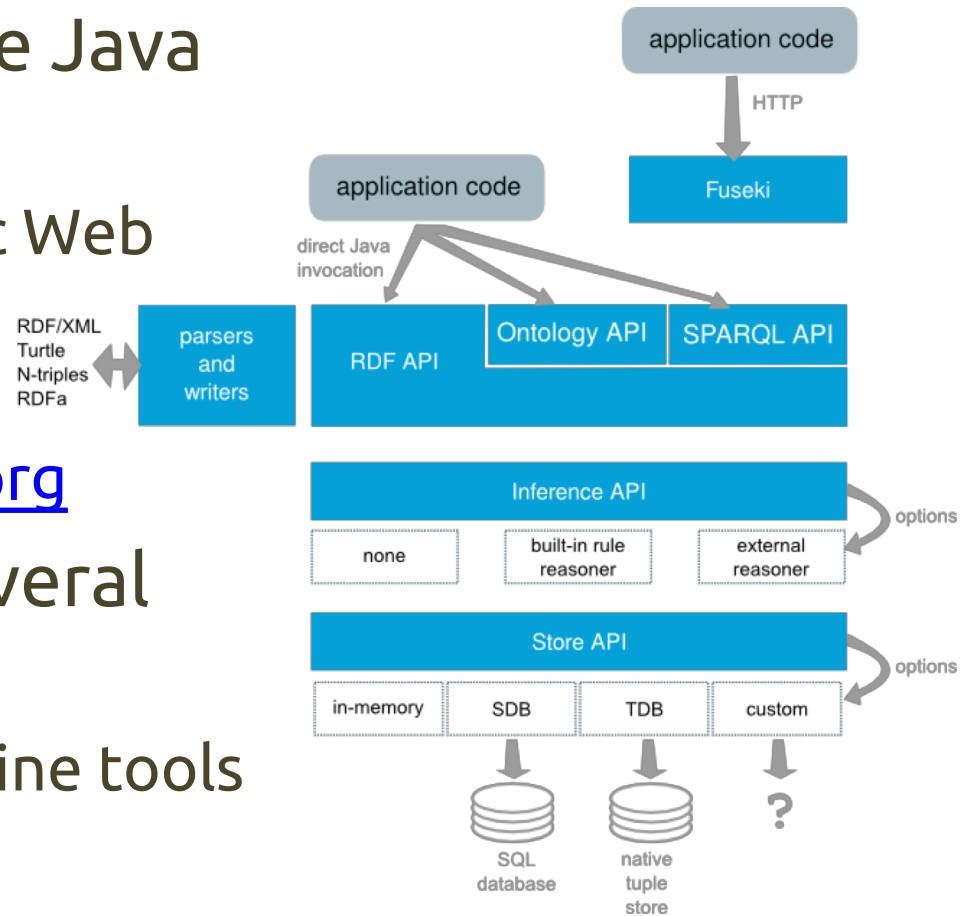
1. Determine the scope
2. Consider reuse
3. Enumerate terms
4. Define classes
5. Define properties
6. Define constraints
7. Create instances

# Now what?

- You created a OWL ontology...
- ... or you want to query some SPARQL endpoints...
- How to do this programmatically?
  - e.g., from a software application
- Good news: frameworks exist!
  - They are written in Java...
  - Apache Jena (RDF/SPARQL/...)
  - OWL API (OWL2)

# Apache Jena

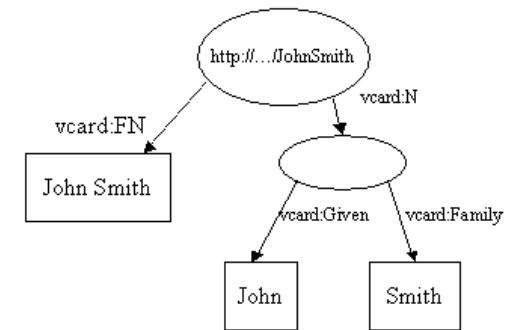
- Free and open source Java framework
  - for building Semantic Web and Linked Data applications
    - <https://jena.apache.org>
- It is composed by several APIs
  - as well as command line tools



# Apache Jena

- Tutorials available
  - <https://jena.apache.org/tutorials/index.html>
  - sample code:  
<https://github.com/apache/jena/tree/master/jena-core/src-examples/jena/examples/rdf>
- It has a limited support to OWL 1.1
  - no OWL2
  - basically, do not use Jena for ontologies!

# Creating a RDF...



```
String personURI = "http://somewhere/JohnSmith";  
String givenName = "John";  
String familyName = "Smith";  
String fullName = givenName + " " + familyName;
```

```
// create an empty model
```

```
Model model = ModelFactory.createDefaultModel();
```

```
// create the resource and add the properties cascading style
```

```
Resource johnSmith = model.createResource(personURI)  
    .addProperty(VCARD.FN, fullName)  
    .addProperty(VCARD.N,  
        model.createResource()  
            .addProperty(VCARD.Given, givenName)  
            .addProperty(VCARD.Family, familyName));
```

# Writing RDF...

- Write the previous model on a `OutputStream`  
`// now write the model in XML form to a file`  
`model.write(System.out);`
- You can also specify the format  
`// you can also specify the format, e.g.,`  
`// model.write(System.out, "TURTLE");`



# Reading RDF...

- Read from a `InputStream`

```
String inputFileName = "vc-db-1.rdf";  
InputStream in =  
FileManager.get().open(inputFileName);  
  
// read the RDF/XML file  
model.read(in, "");
```

The base URI to be used when converting  
relative URI's to absolute URI's

# SPARQL

- Jena supports SPARQL querying through the ARQ engine
  - Standard SPARQL
  - Free text search via Lucene
  - Access and extension of the SPARQL algebra
  - Property functions for custom processing of semantic relationships
  - Aggregation, GROUP BY and assignment as SPARQL extensions
  - Client-support for remote access to any SPARQL endpoint
  - ...

# SPARQL with ARQ

[...]

```
// Create a new query
String queryString =
    "PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
    "SELECT ?url " +
    "WHERE {" +
    "    ?contributor foaf:name \"Luigi De Russis\" . " +
    "    ?contributor foaf:weblog ?url . " +
    "    }";

Query query = QueryFactory.create(queryString);

// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();

// Output query results
ResultSetFormatter.out(System.out, results, query);

// Free up resources used running the query
qe.close();
```

# OWL API

- A Java API and reference implementation
  - for creating, manipulating and serializing OWL 2 Ontologies
  - <http://owlcs.github.io/owlapi>
- Free and open source
- Created and maintained by the University of Manchester
  - <http://owl.cs.manchester.ac.uk>

# OWL API

- It includes the following components
  - API for OWL 2 and an efficient in-memory reference implementation
  - RDF/XML parser and writer
  - OWL/XML parser and writer
  - OWL Functional Syntax parser and writer
  - Turtle parser and writer
  - SWRL
  - Reasoner interfaces
    - towards, e.g., FaCT++, Hermit, Pellet, and Racer

# OWL API

- Documentation and Javadocs
  - <https://github.com/owlcs/owlapi/wiki>
  - [http://owlcs.github.io/owlapi/apidocs\\_4/index.html](http://owlcs.github.io/owlapi/apidocs_4/index.html)
  - scarce and not updated, sometimes
- Versions
  - 5.0, cutting edge, Java 8 only
  - 4.0, stable, Java 7+
    - currently used by Protégé
    - several examples are available, right now

# OWL API Fundamentals

- `OWLOntology`
  - an interface
  - modelling a set of logical and nonlogical `OWLAxioms`, with a name (an `IRI`) and convenience methods to retrieve such axioms
- `OWLEntity`
  - anything that can be identified with an `IRI`, i.e., class names, data and object properties, named individuals, ...

# OWL API Fundamentals

- `OWLAxiom`
  - the basic unity
  - TBox axioms describe relations between classes and class expressions (equivalence, subsumption, disjointness)
  - ABox axioms (assertions) describe relations between individuals and between individuals and classes/class expressions
  - RBox axioms describe relations between properties



# Load (or create) an ontology...

- **Ontology creation**

```
OWLOntologyManager m =  
    OWLManager.createOWLOntologyManager();  
OWLOntology o = m.createOntology(example_iri);
```

- **Ontology loading**

```
OWLOntologyManager m =  
    OWLManager.createOWLOntologyManager();  
OWLOntology o =  
    m.loadOntologyFromOntologyDocument(ont_iri);
```

 a File or a IRI

# Save an ontology...

- Save in OWL/XML format

```
m.saveOntology(ontology, new  
OWLXMLOntologyFormat(), file);
```

- Save in RDF/XML format

```
m.saveOntology(ontology, file);
```

# Example

- It uses the University ontology we developed last week to
  - load the ontology
  - compute logical inferences
  - ask for
    - university individuals
    - which degrees each university offers
    - which courses each degree offers
    - who is enrolled in those courses
- Get it from
  - <https://github.com/luigidr/owlapi-example>

# Example Requirements

- OWL API 4.5.4
- HermiT reasoner 1.3.8.413
  - download from <https://github.com/owlcs/owlapi/wiki/Reasoners,-OWL-API-Support,-papers-about-the-OWL-API>
- Gradle
  - for handling the OWL API/HermiT set up and dependencies
  - <https://gradle.org>

# Questions?




**01RRDIU SEMANTIC WEB**

Luigi De Russis

luigi.derussis@polito.it



# License

- This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
- You are free:
  - to **Share** - to copy, distribute and transmit the work
  - to **Remix** - to adapt the work
- Under the following conditions:
  -  **Attribution** - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  -  **Noncommercial** - You may not use this work for commercial purposes.
  -  **Share Alike** - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit <http://creativecommons.org/license/by-nc-sa/3.0/>