
Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

# Developing Smart Java™ Code with Semantic Web Technology

Holger Knublauch

TopQuadrant, Inc.

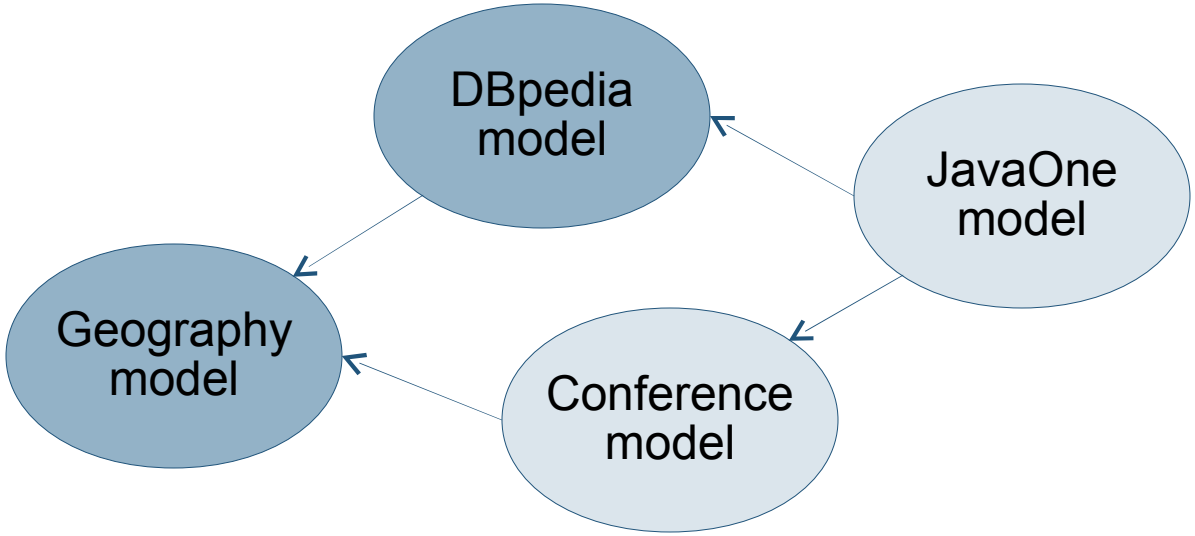
holger@topquadrant.com

TopQuadrant™

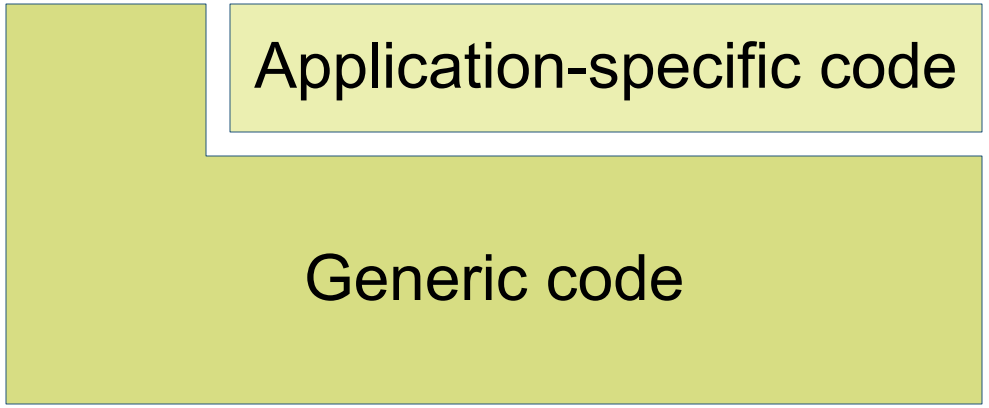
W3C® MEMBER

# Semantic Web Elevator Pitch

Linked Smart Data




Smart Code



# Overview of this talk

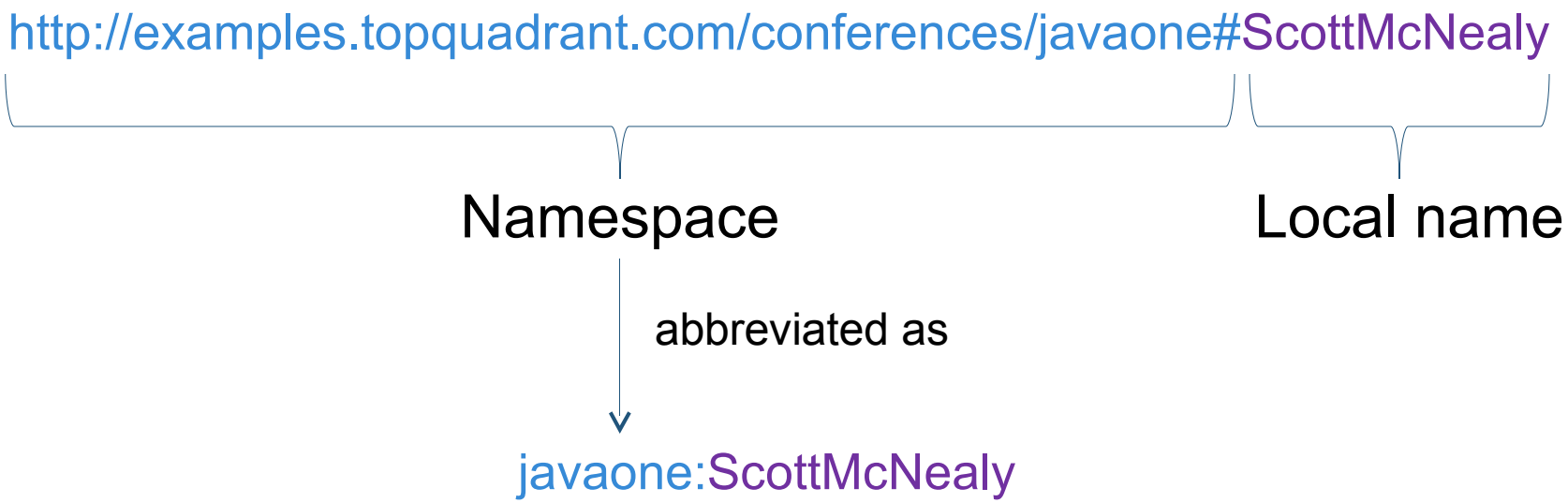
- Introduction to RDF
  - Resources, literals and triples
- Linked Data
- RDF Schema
  - Classes, properties, and instances
- Java APIs and Tools
- Data Binding and software architecture
- SPARQL
  - Queries and rules
- Model-driven applications

# Semantic Web Technology

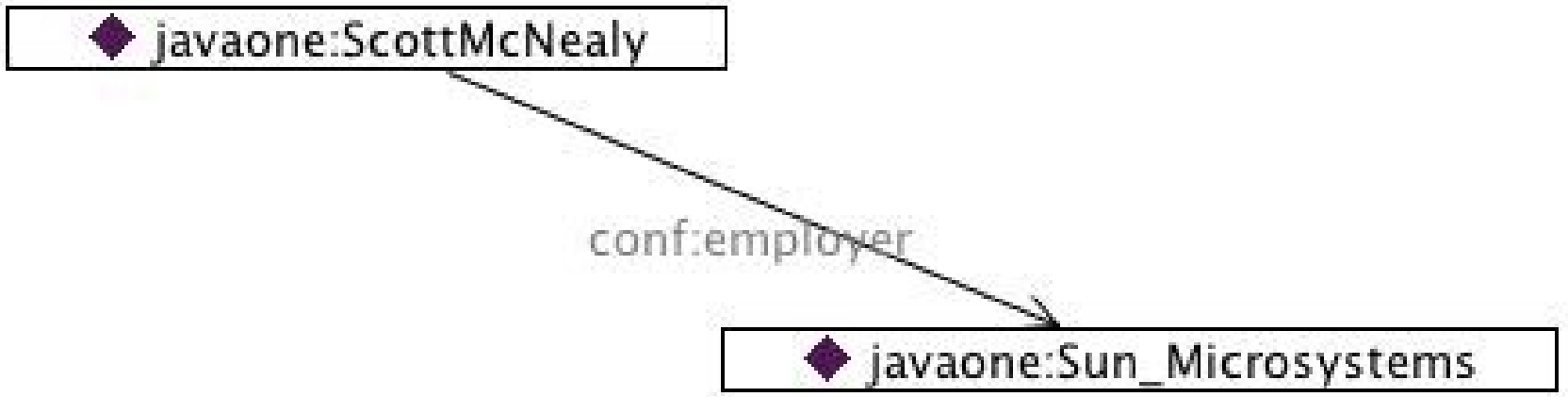
- Infrastructure to link data on the Web
- Global database
- Borrows some ideas of object-oriented languages
- Data is self-describing
- W3C standards
  - RDF
  - RDF Schema and OWL
  - SPARQL
  - RDFa, GRDDL

# RDF: Resources

- Semantic Web entities have a unique identifier

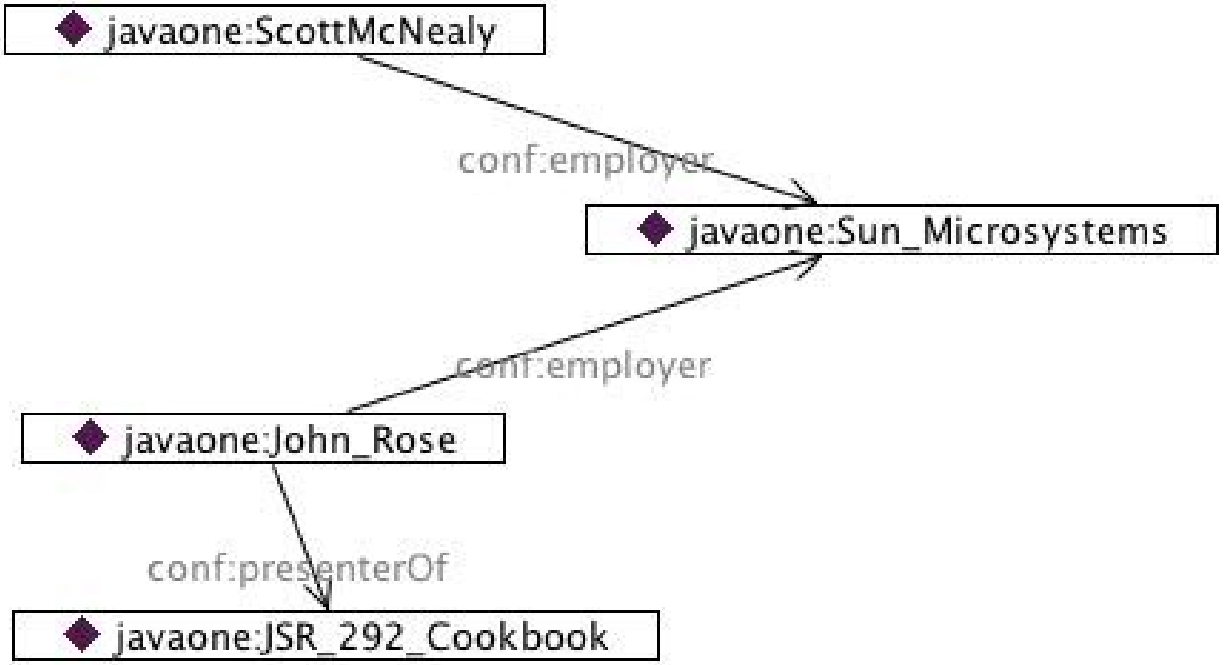


# RDF: Triples



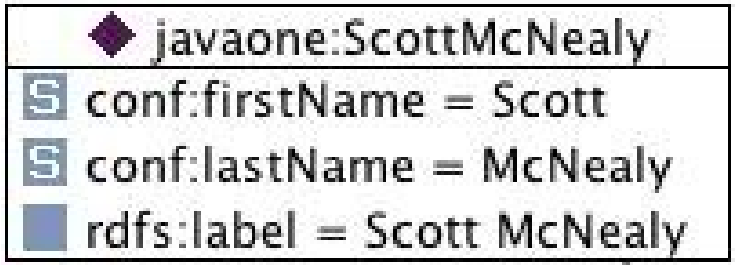

| Subject | Predicate | Object |
|---|---|---|
| javaone:ScottMcNealy | conf:employer | javaone:Sun_Microsystems |

# RDF: Graphs



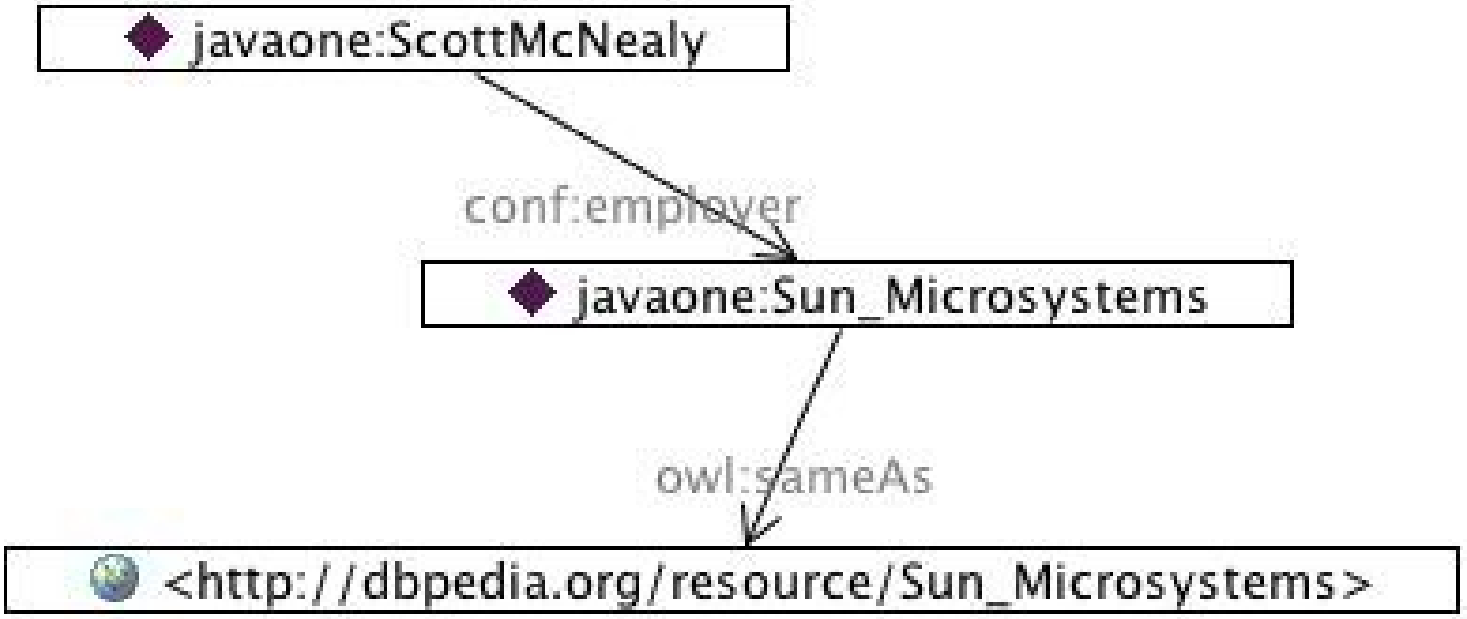

| Subject | Predicate | Object |
|---|---|---|
| javaone:ScottMcNealy | conf:employer | javaone:Sun_Microsystems |
| javaone:John_Rose | conf:employer | javaone:Sun_Microsystems |
| javaone:John_Rose | conf:presenterOf | javaone:JSR_292_Cookbook |

# RDF: Datatype property values

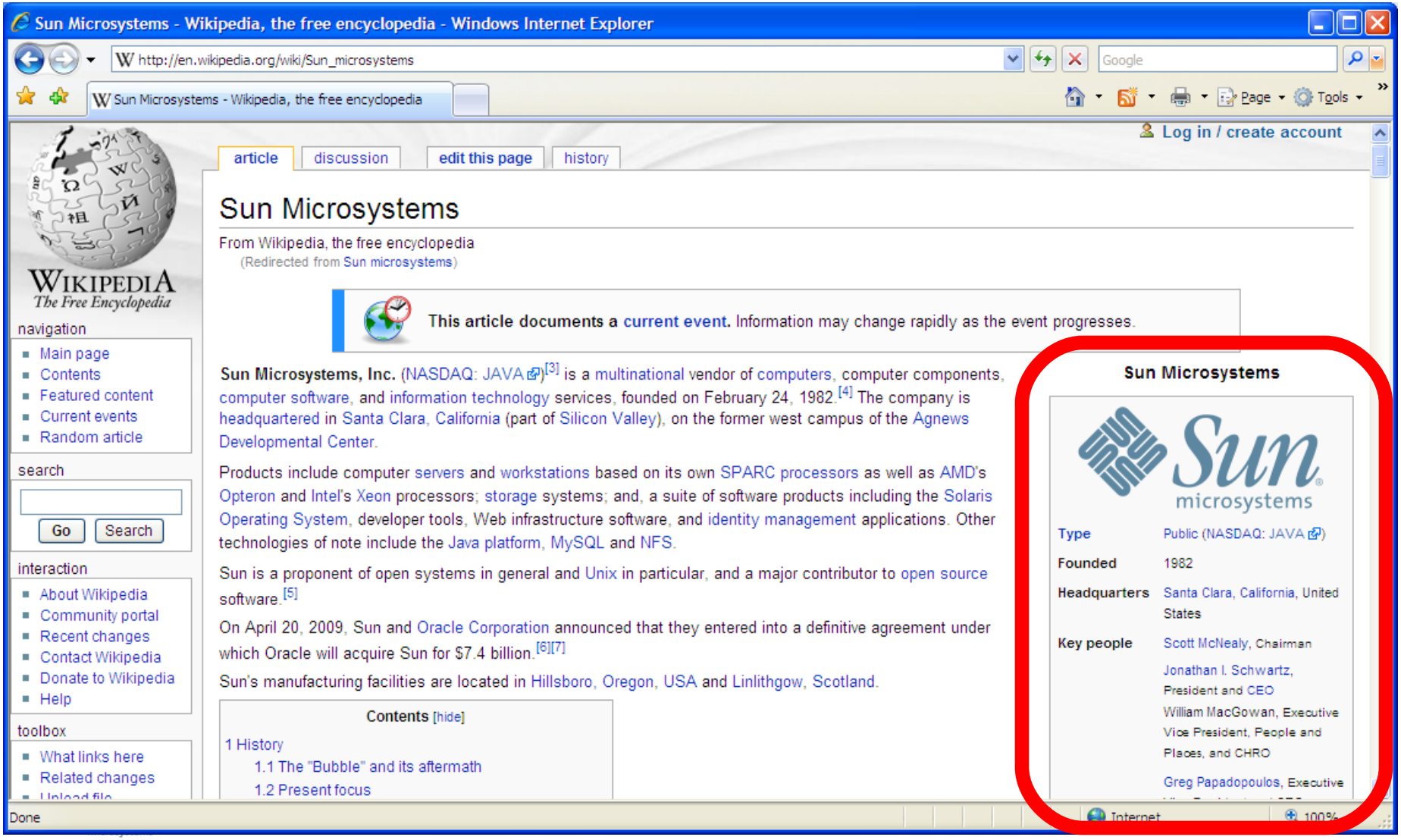



| Subject | Predicate | Object |
|---|---|---|
| javaone:ScottMcNealy | conf:firstName | “Scott”^^xsd:string |
| javaone:ScottMcNealy | conf:lastName | “McNealy”^^xsd:string |
| javaone:ScottMcNealy | rdfs:label | “Scott McNealy” |

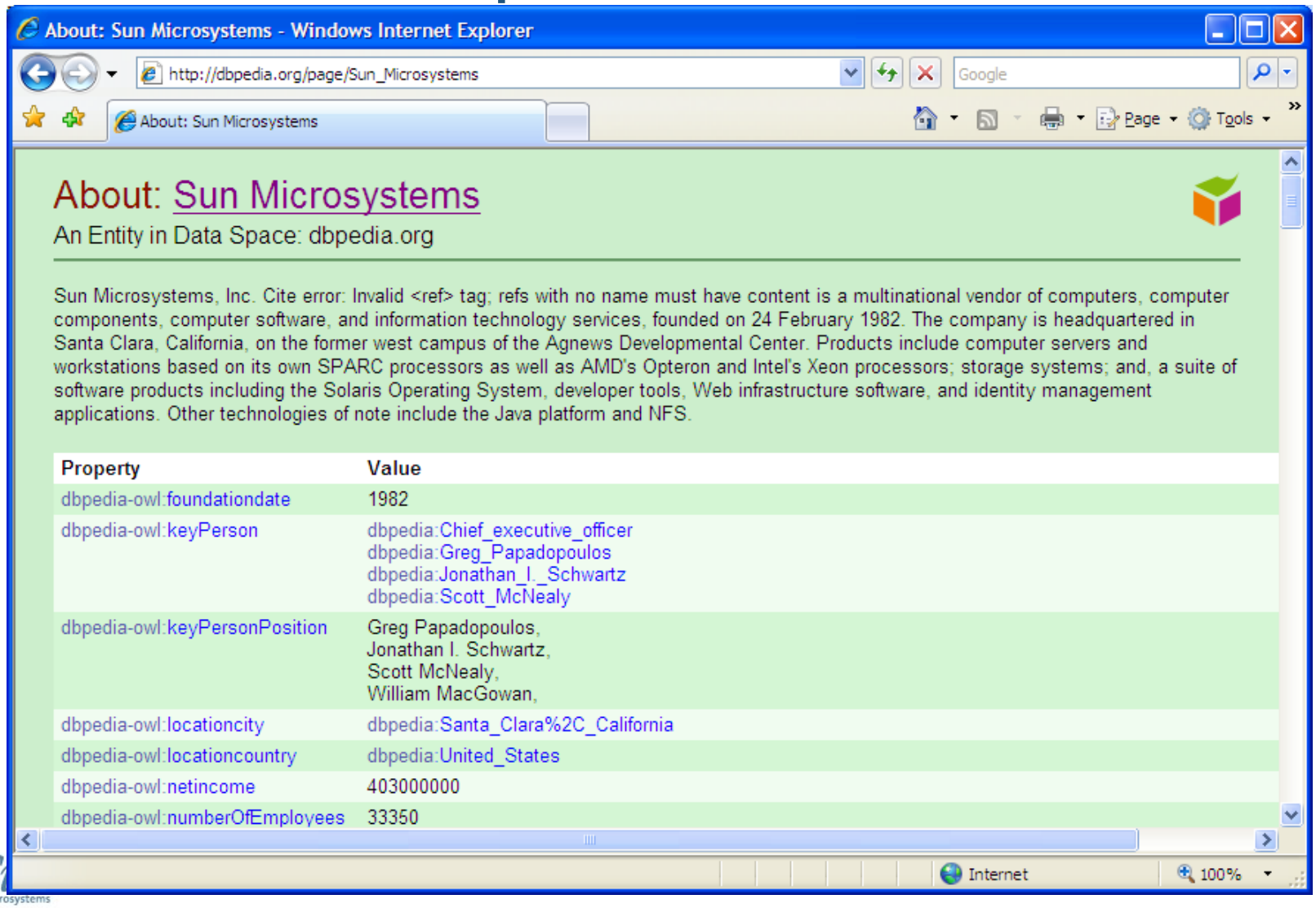
# Linked Data


javaone:ScottMcNealy
conf:employer
javaone:Sun_Microsystems
owl:sameAs
<http://dbpedia.org/resource/Sun_Microsystems>

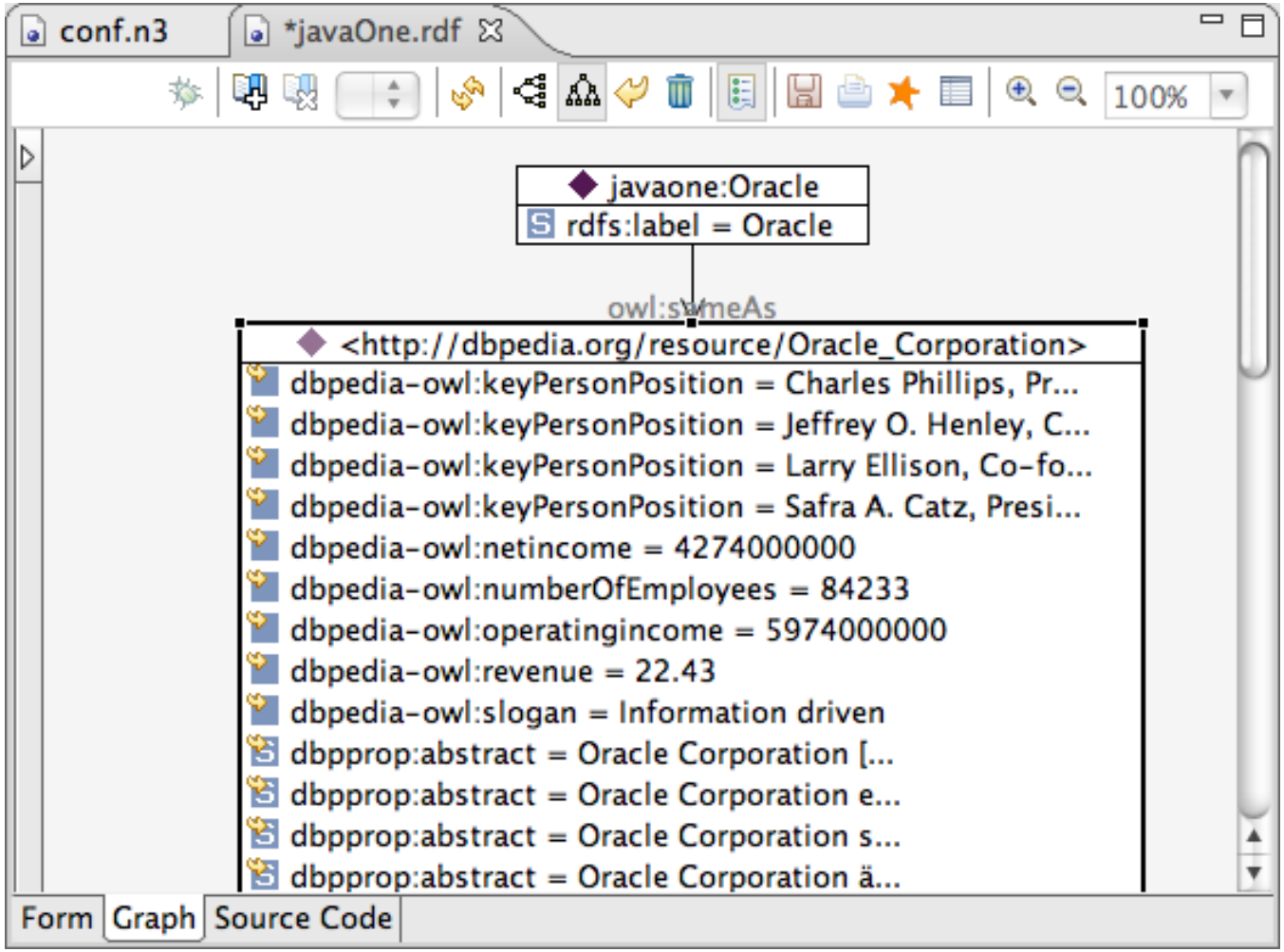
# Linked Data: Wikipedia



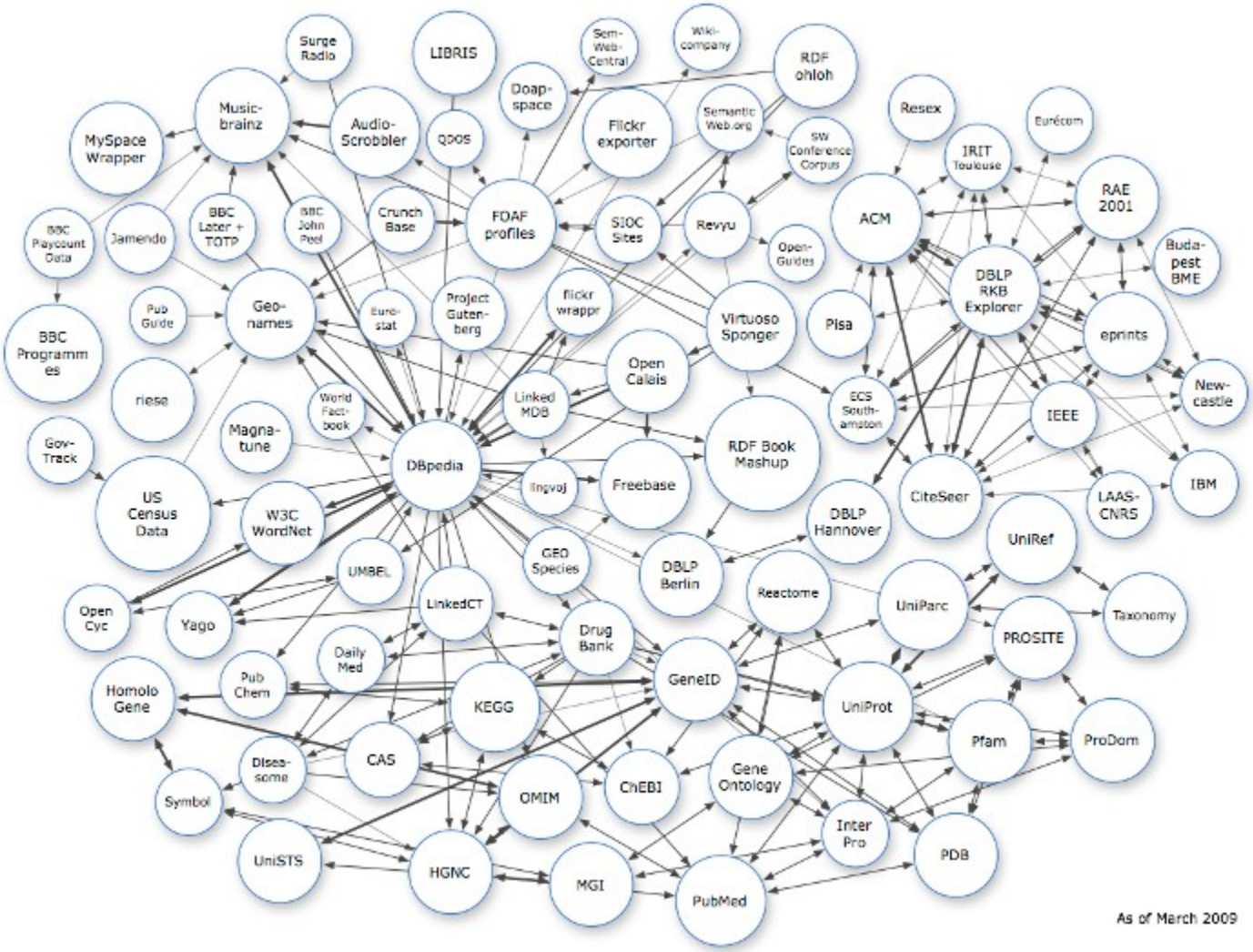
# Linked Data: DBpedia Browser



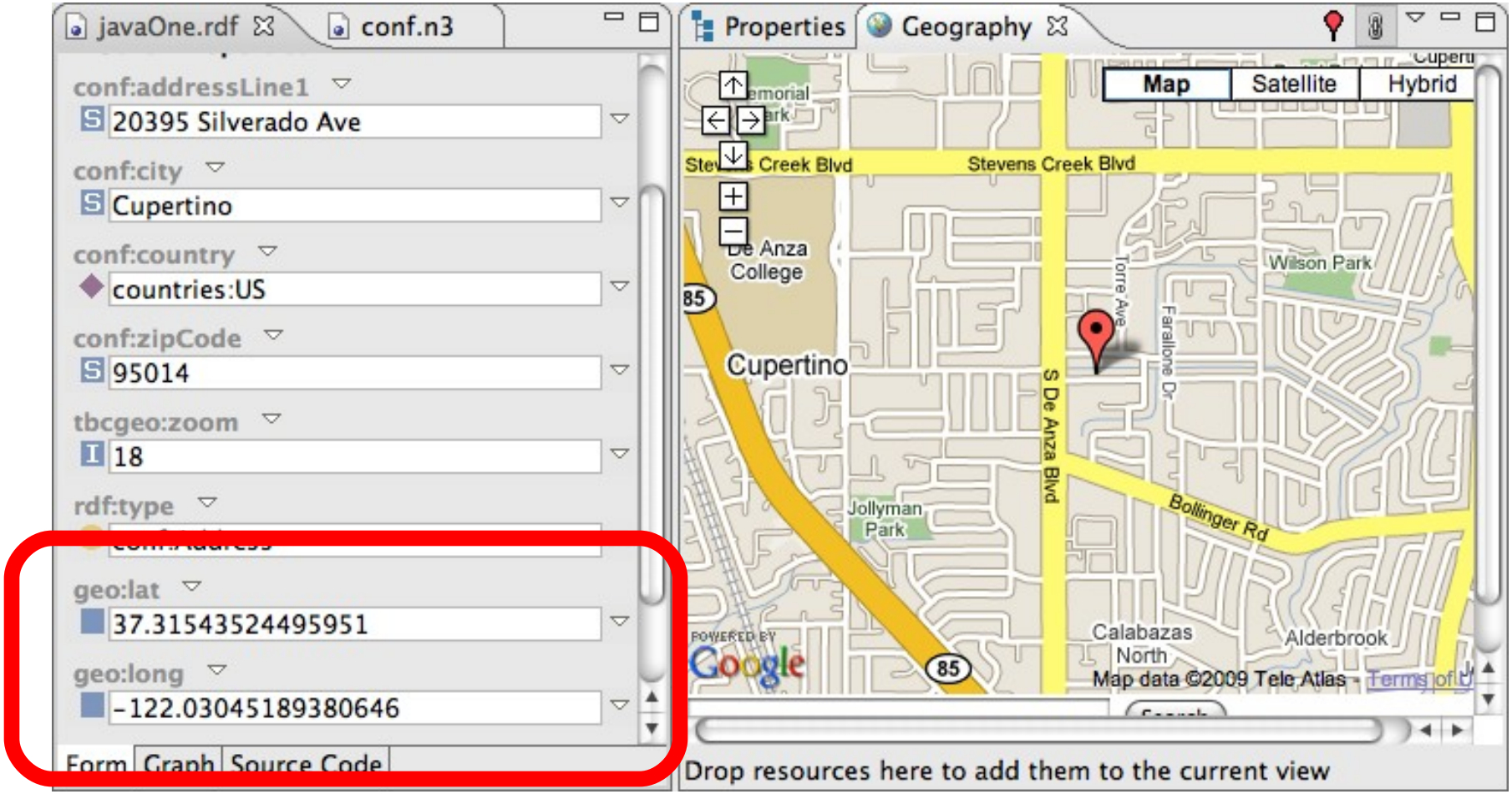
# Linked Data: more RDF triples, for free!



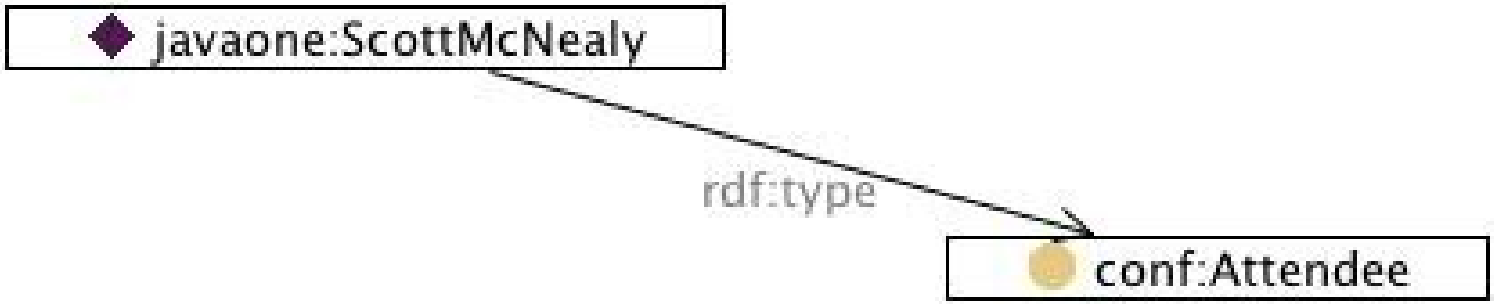
# Linked Data: join the cloud



# Linked Data: Geography ontology

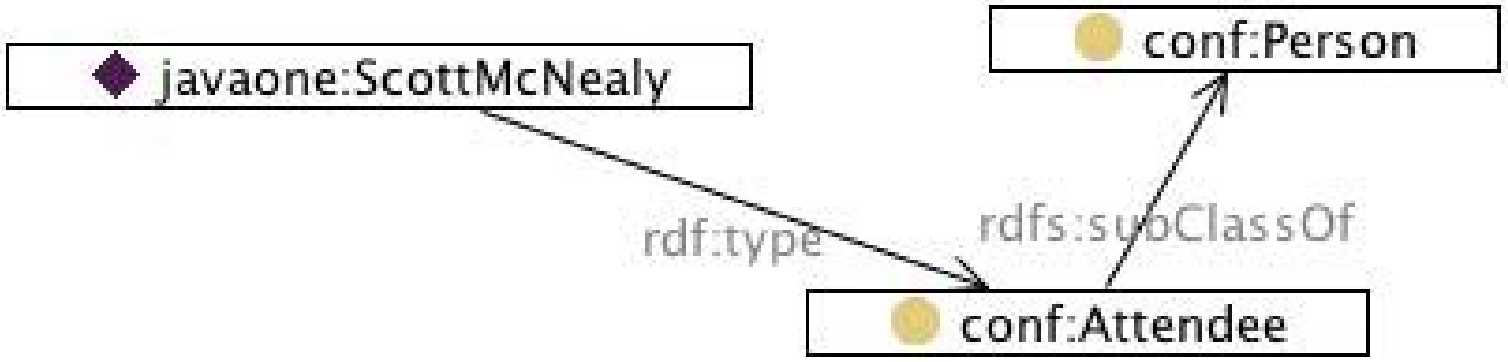


# RDF Schema: Type relationships



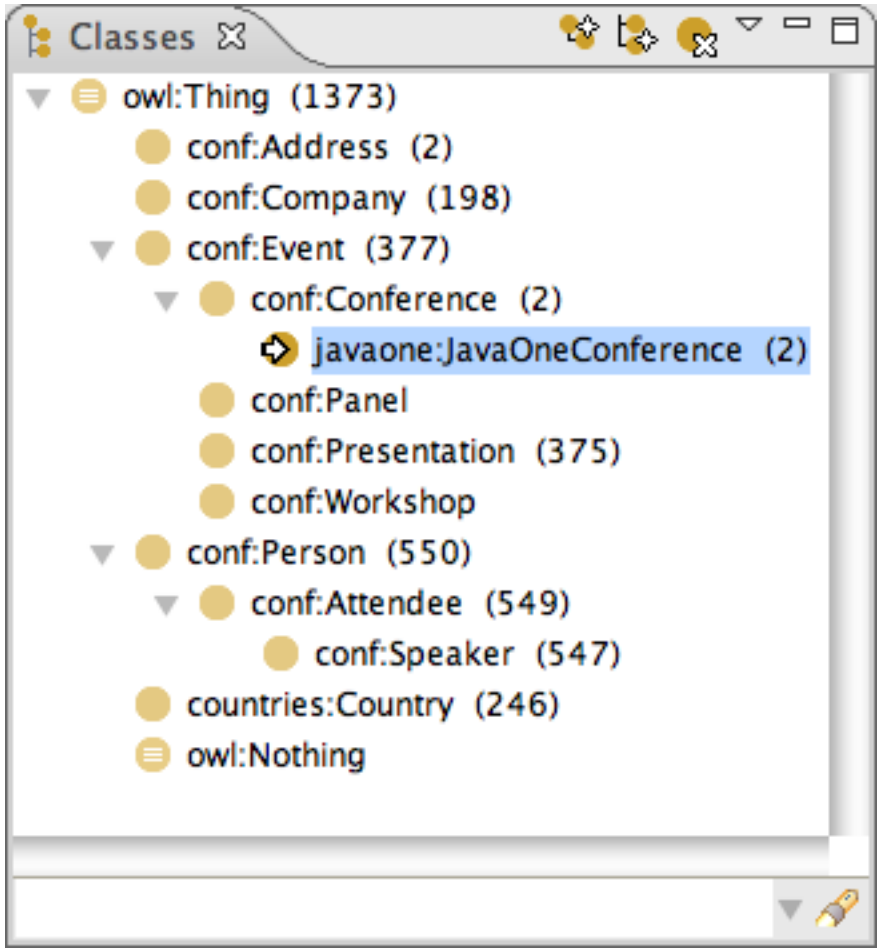

| Subject | Predicate | Object |
|---|---|---|
| javaone:ScottMcNealy | rdf:type | conf:Attendee |

# RDF Schema: Subclass relationships

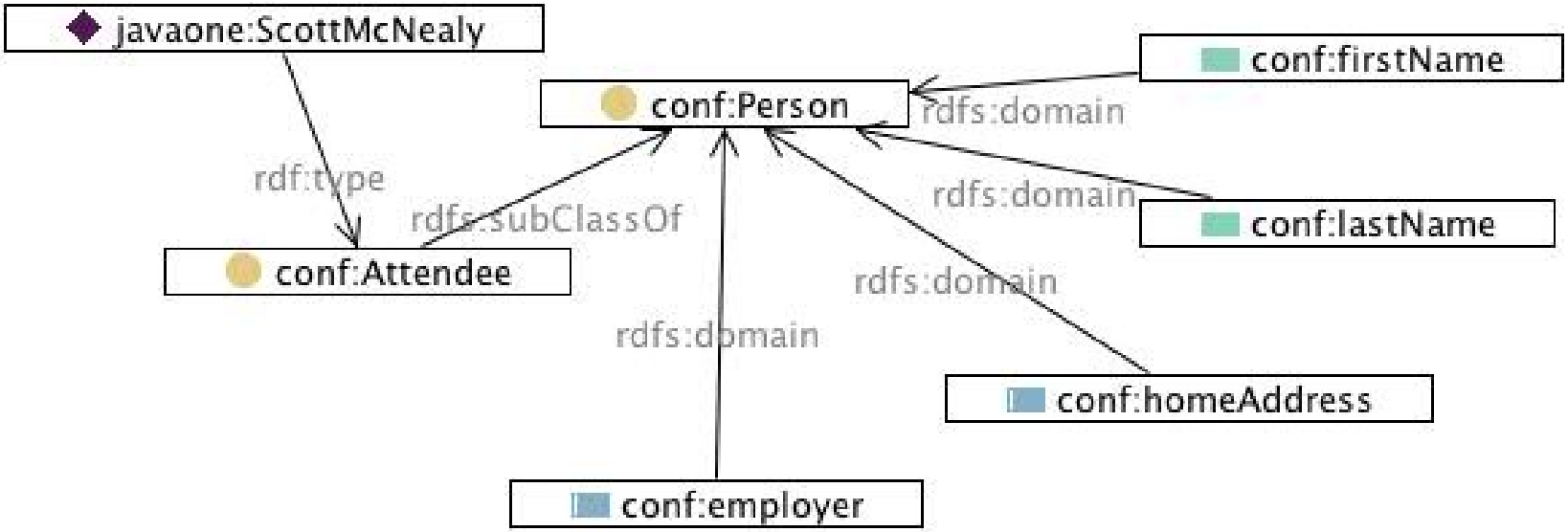



| Subject | Predicate | Object |
|---|---|---|
| javaone:ScottMcNealy | rdf:type | conf:Attendee |
| conf:Attendee | rdfs:subClassOf | conf:Person |

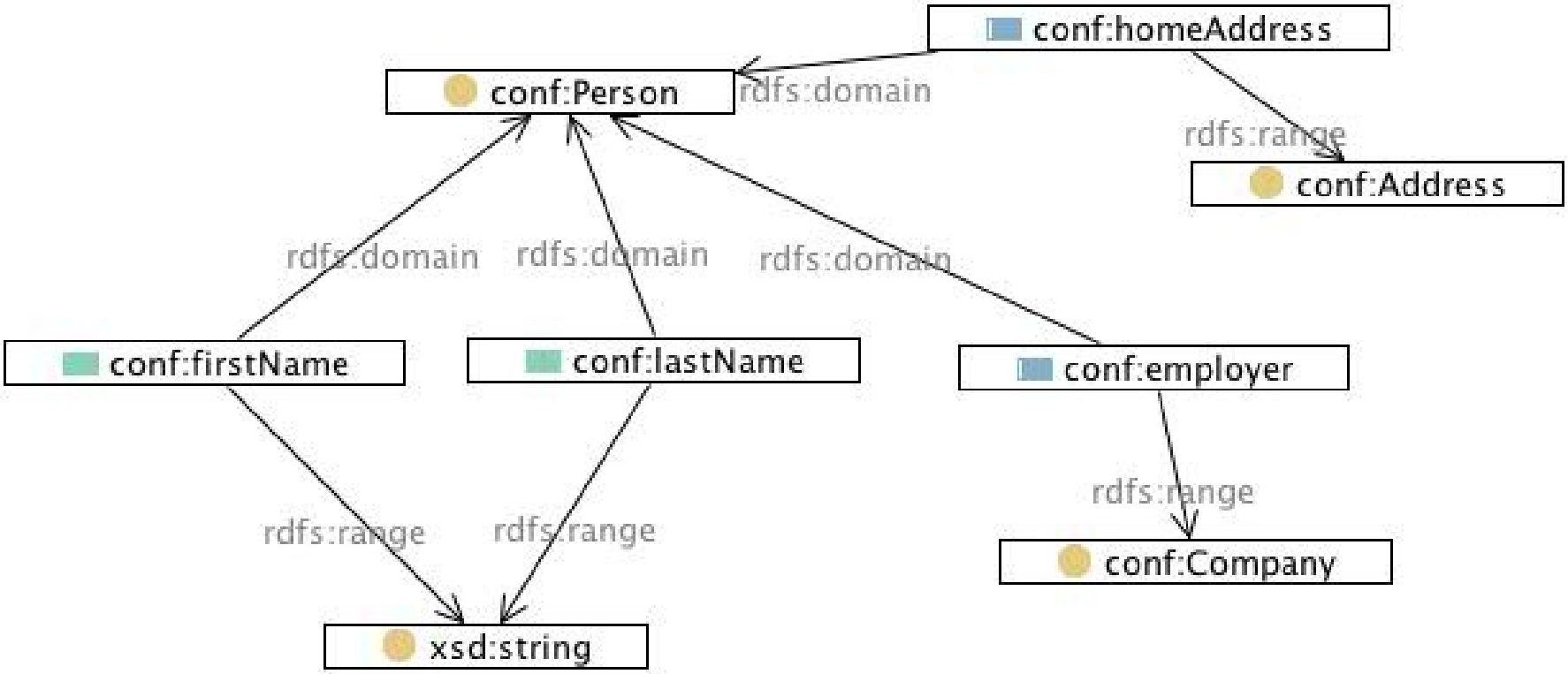
# RDF Schema: Class Hierarchy



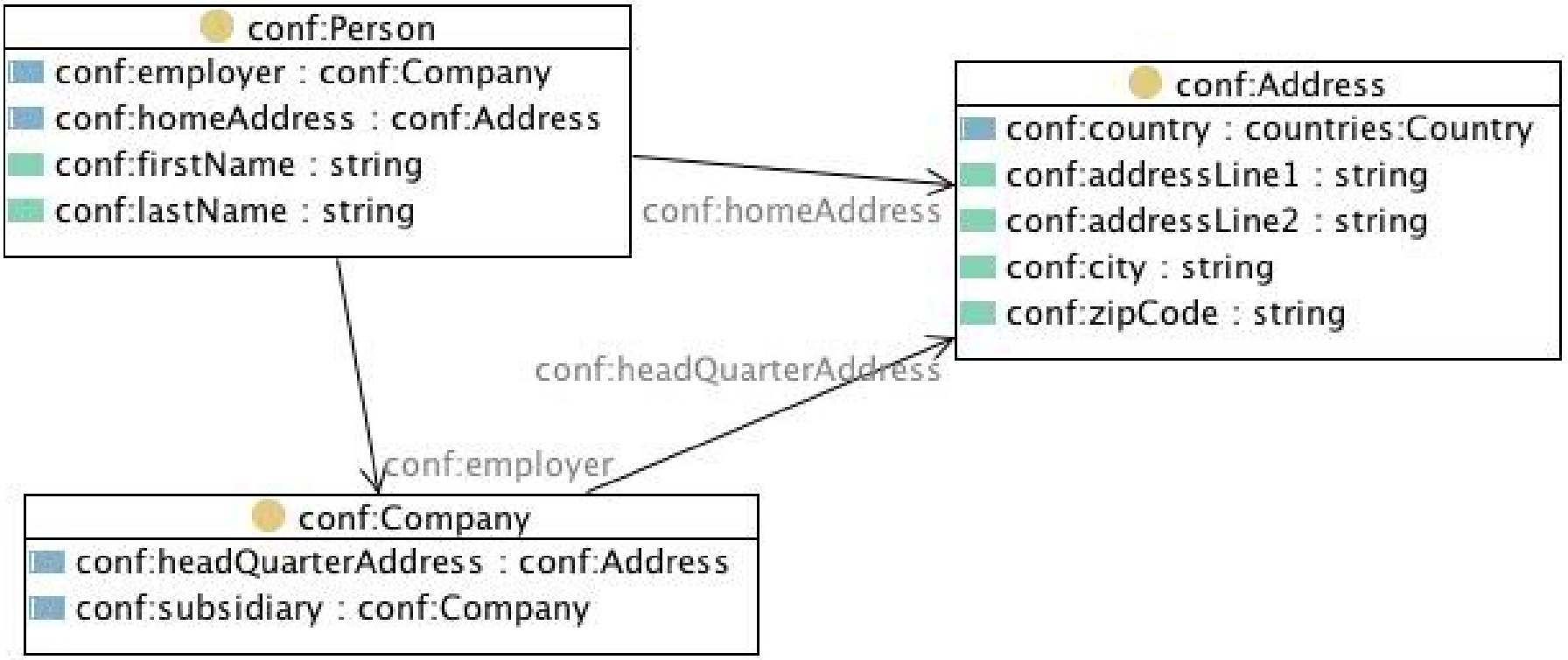
# RDF Schema: Property Domains



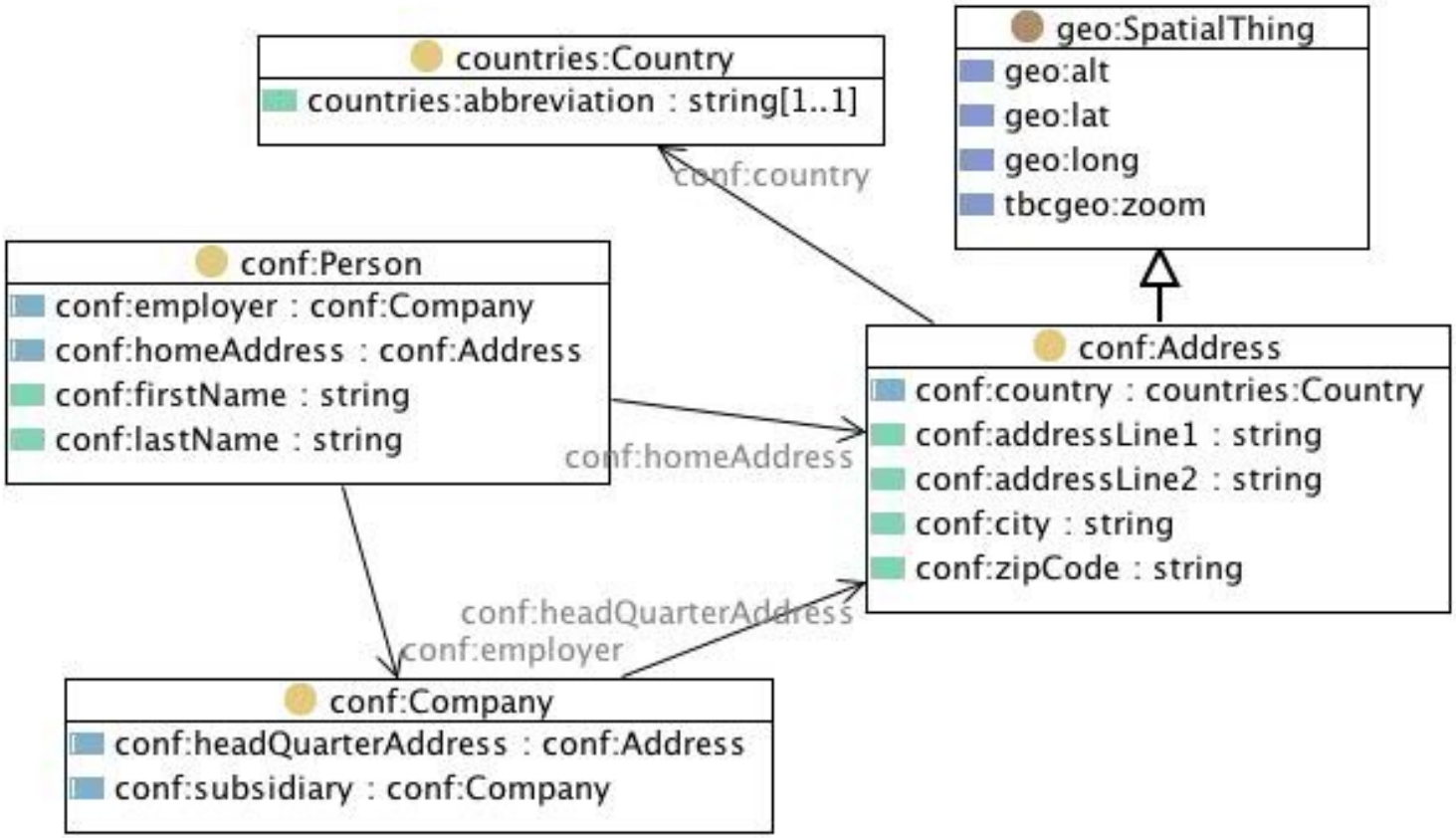
# RDF Schema: Property Ranges



# RDF Schema: from a UML point of view



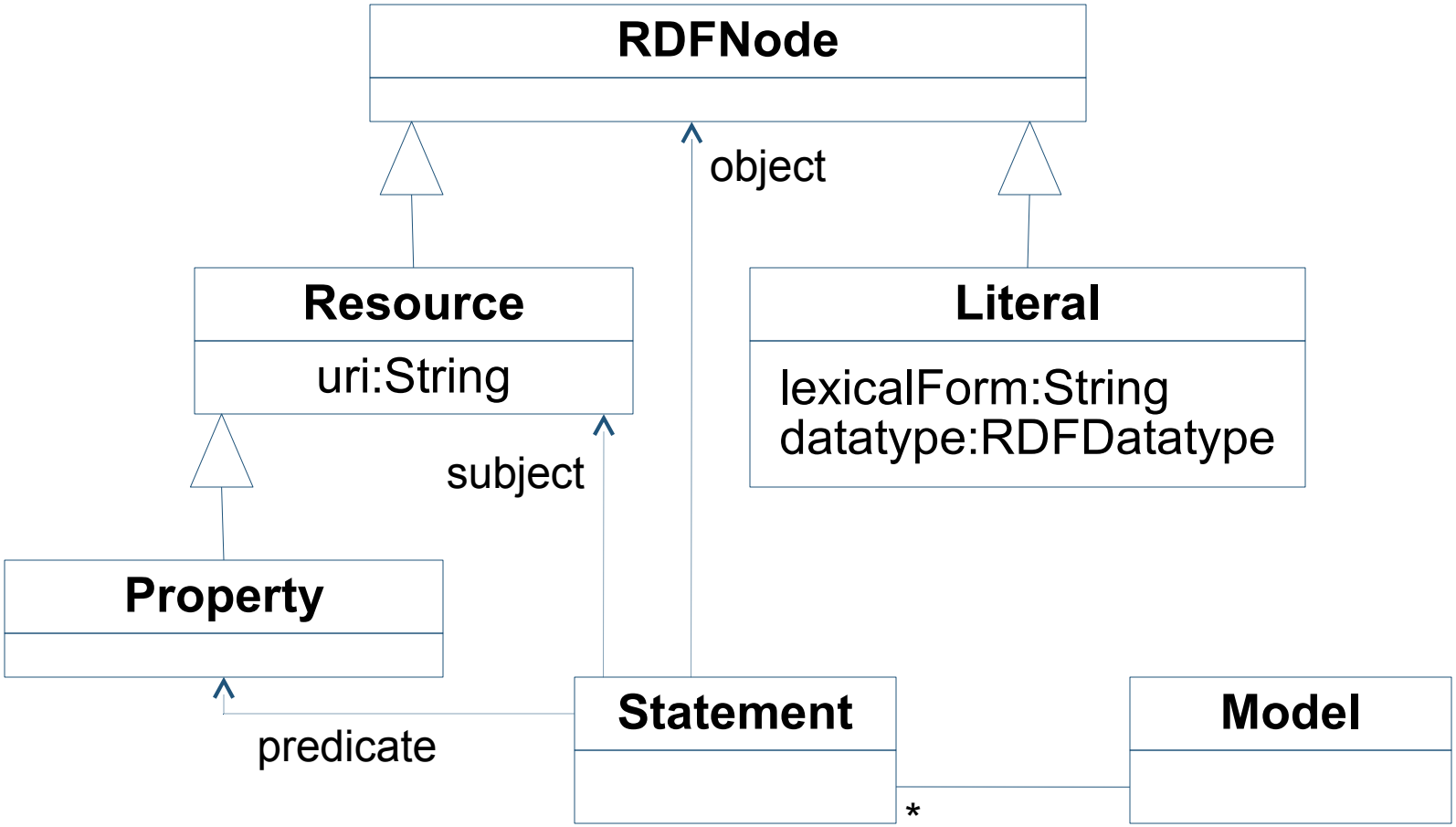
# RDF Schema: linking namespaces



# General Java APIs for RDF

- Jena
  - http://jena.sourceforge.net/
- Sesame
  - http://www.openrdf.org/

# Jena API: Core interfaces (simplified)



# Jena API: “Hello, World”

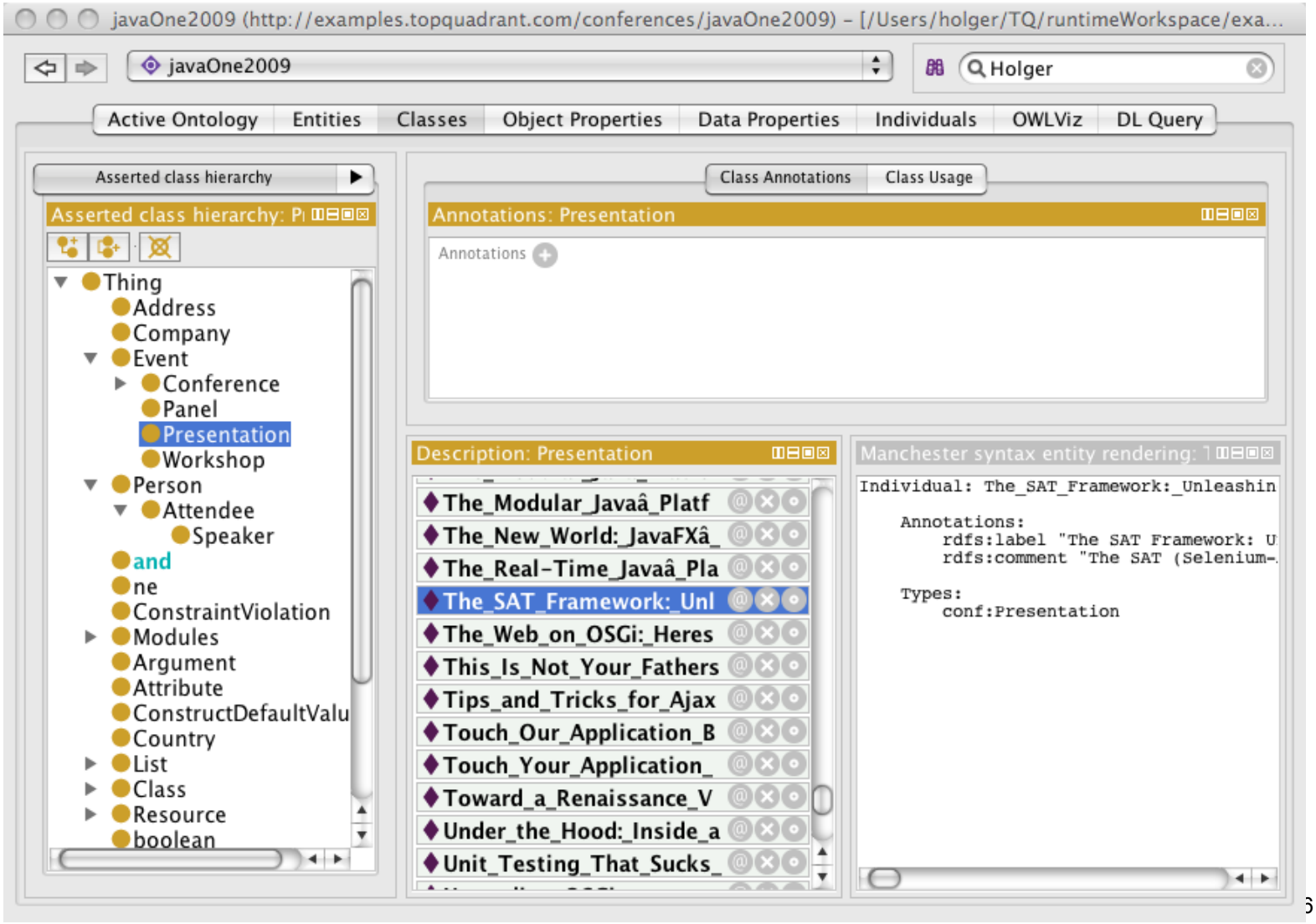
```
// Create a Jena Model and init some namespace prefixes
String NS = "http://examples.topquadrant.com/conferences/conf#";
Model model = ModelFactory.createDefaultModel();
model.setNsPrefix("conf", NS);
model.setNsPrefix("rdfs", RDFS.getURI());
model.setNsPrefix("xsd", XSD.getURI());

// Create an example triple
Resource javaOne = model.createResource(NS + "JavaOne2009");
Literal label = model.createTypedLiteral("JavaOne 2009");
model.add(javaOne, RDFS.label, label);

// Write the Model in N3 format to the screen
model.write(System.out, FileUtils.langN3);
```

# Jena API: “Hello, World” Output

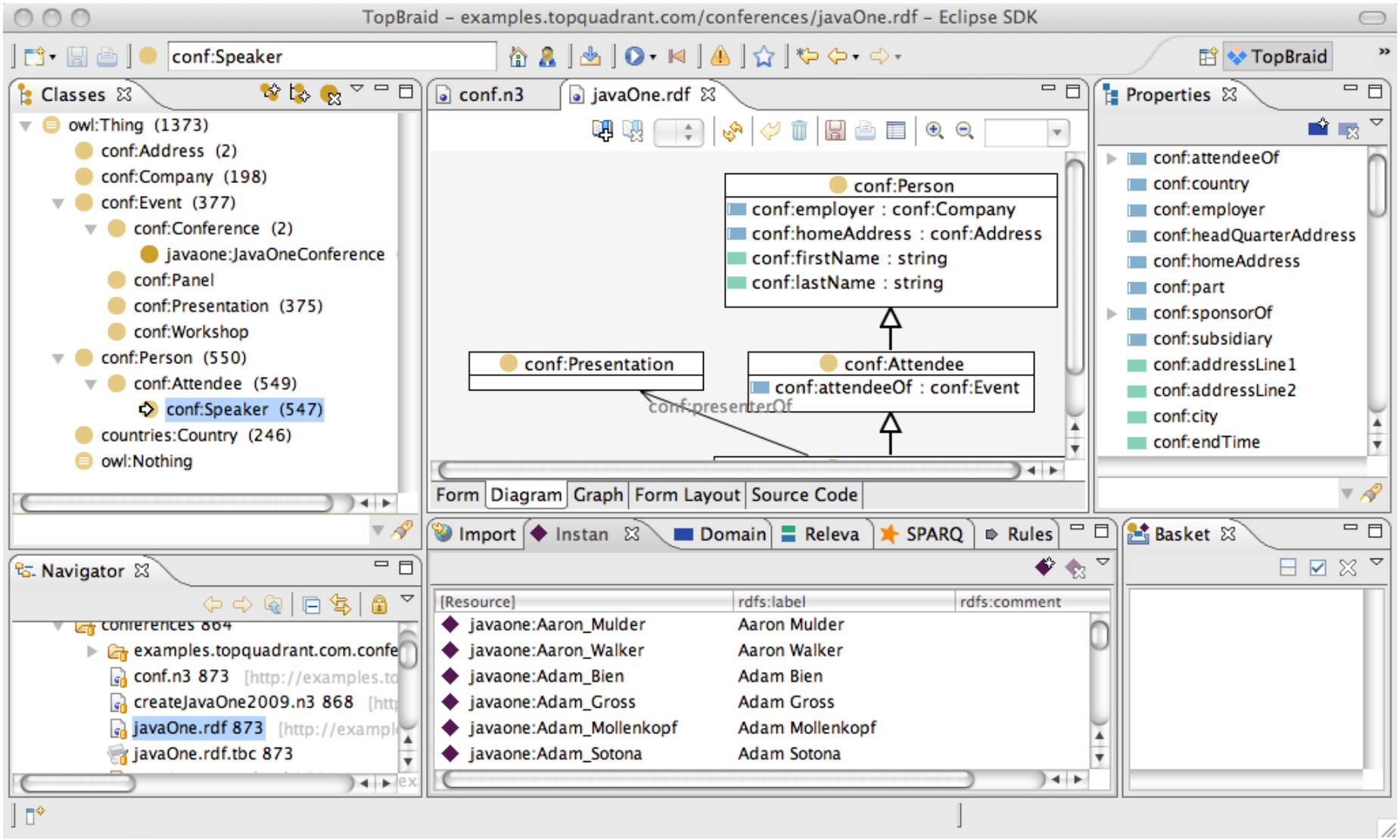
```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix conf:    <http://examples.topquadrant.com/conferences/conf#> .

conf:JavaOne2009
     rdfs:label "JavaOne 2009"^^xsd:string .
```

RDF file in N3 format
Options include RDF/XML

# Tools: OWL ontology editor Protege



# Tools: TopBraid Composer



# Development Methodology

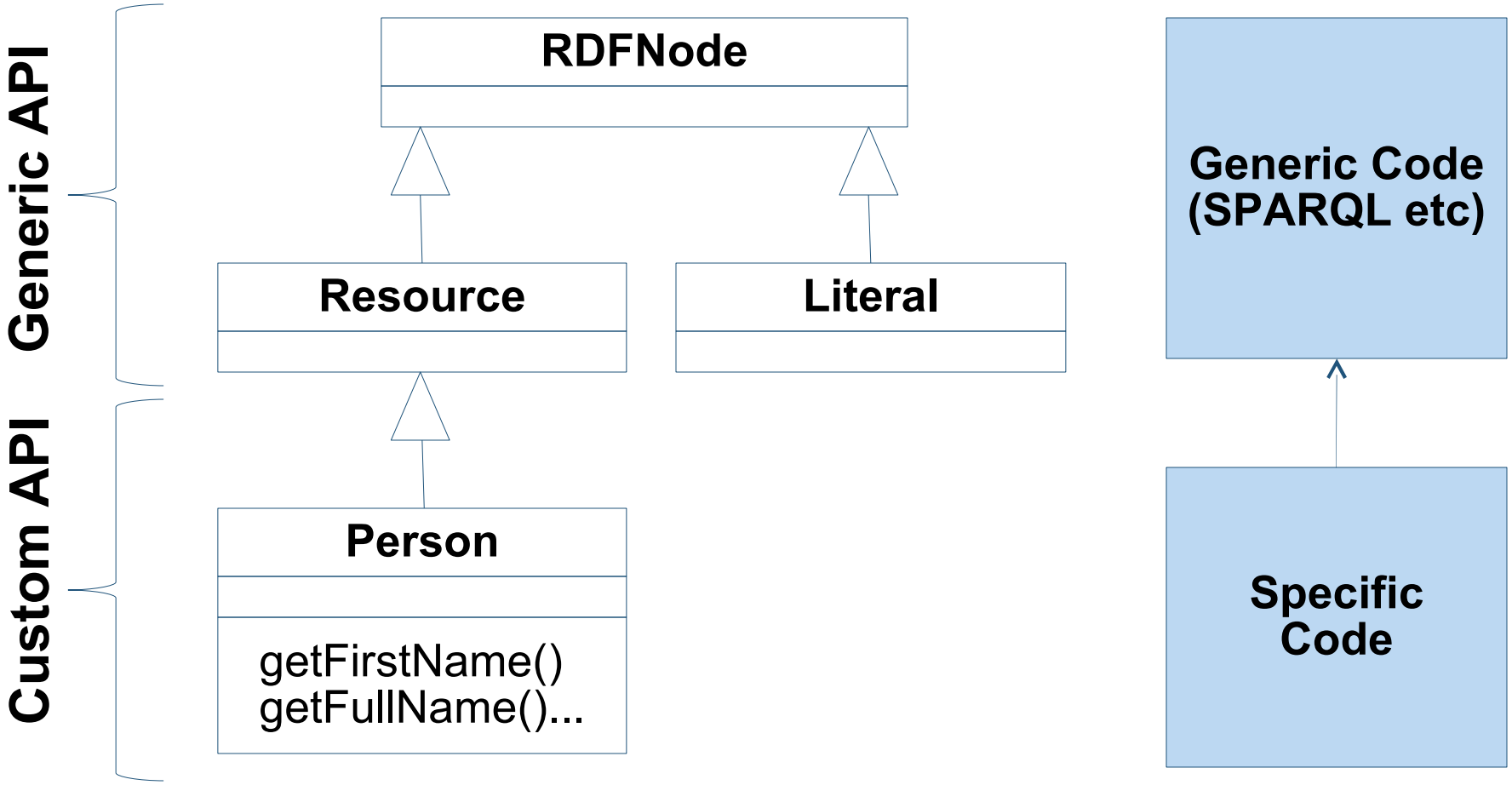
- Define/re-use schema
- Play with example instances, queries etc
- Take a data-centric approach!
- Link key classes of the schema to your Java code (Data Binding)
- Assume schema evolution
- Develop smart, generic code instead of domain-specific code

# Data Binding: Generated schema class

CONF.java

```
package com.topquadrant.examples.conf.vocabulary;

import com.hp.hpl.jena.rdf.model.Property;

/**
 * Vocabulary for http://examples.topquadrant.com/conferences/conf
 */
public class CONF {

    public final static String BASE_URI = "http://examples.topquadrant.com/conferences/conf";

    public final static String NS = BASE_URI + "#";

    public final static String PREFIX = "conf";


    public final static Resource Address = ResourceFactory.createResource(NS + "Address");

    public final static Resource Attendee = ResourceFactory.createResource(NS + "Attendee");

    public final static Resource Company = ResourceFactory.createResource(NS + "Company");

    public final static Resource Conference = ResourceFactory.createResource(NS + "Conference");
```

# Data Binding: Basic Architecture

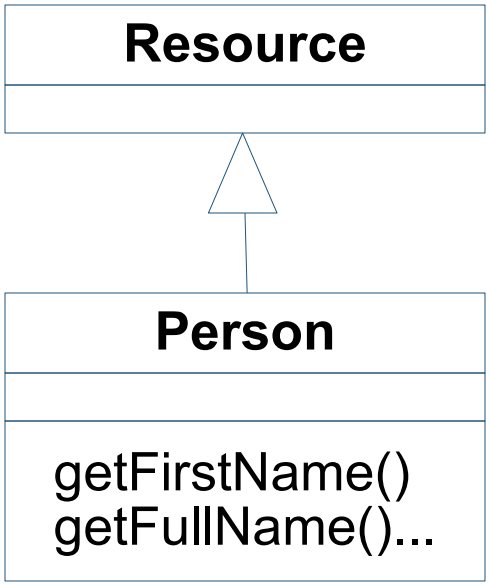


# Data Binding: Example use

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

// Create a new Person instance
String steveURI = CONF.NS + "SteveJobs";
Person steve = ConfFactory.createPerson(model, steveURI);

// Set some property values
steve.setFirstName("Steve");
steve.setLastName("Jobs");

// Query some property values
String fullName = steve.getFullName();
System.out.println("Person: " + fullName);
```

# Data Binding: Example interface

```
public interface Person extends Resource {

    String getFirstName();

    String getFullName();

    String getLastName();

    void setFirstName(String value);

    void setLastName(String value);
}
```



# Data Binding: Example implementation

```
public class PersonImpl extends ResourceImpl implements Person {

    public PersonImpl(Node node, EnhGraph eg) {
        super(node, eg);
    }

    public String getFirstName() {
        return getStringProperty(CONF.firstName);
    }

    public String getFullName() {
        return getFirstName() + " " + getLastName();
    }
```

# Data Binding: Factory class

```
public class ConfFactory {
    static {
        register(Person.class, PersonImpl.class);
    }
    public static Person createPerson(Model model, String uri) {
        return model.createResource(uri, CONF.Person).as(Person.class);
    }
    public static Person getPerson(Model model, String uri) {
        return model.createResource(uri).as(Person.class);
    }
...
```

# Data Binding: Frameworks

- ELMO
  - http://www.openrdf.org/doc/elmo/1.4/
- Sommer
  - https://sommer.dev.java.net/sommer/index.html
- JenaBean
  - http://code.google.com/p/jenabean/
  - http://www.incunabulum.de/projects/it/owl2java/
- Kazuki
  - http://projects.semwebcentral.org/projects/kazuki/

# Data Binding: Java code generators

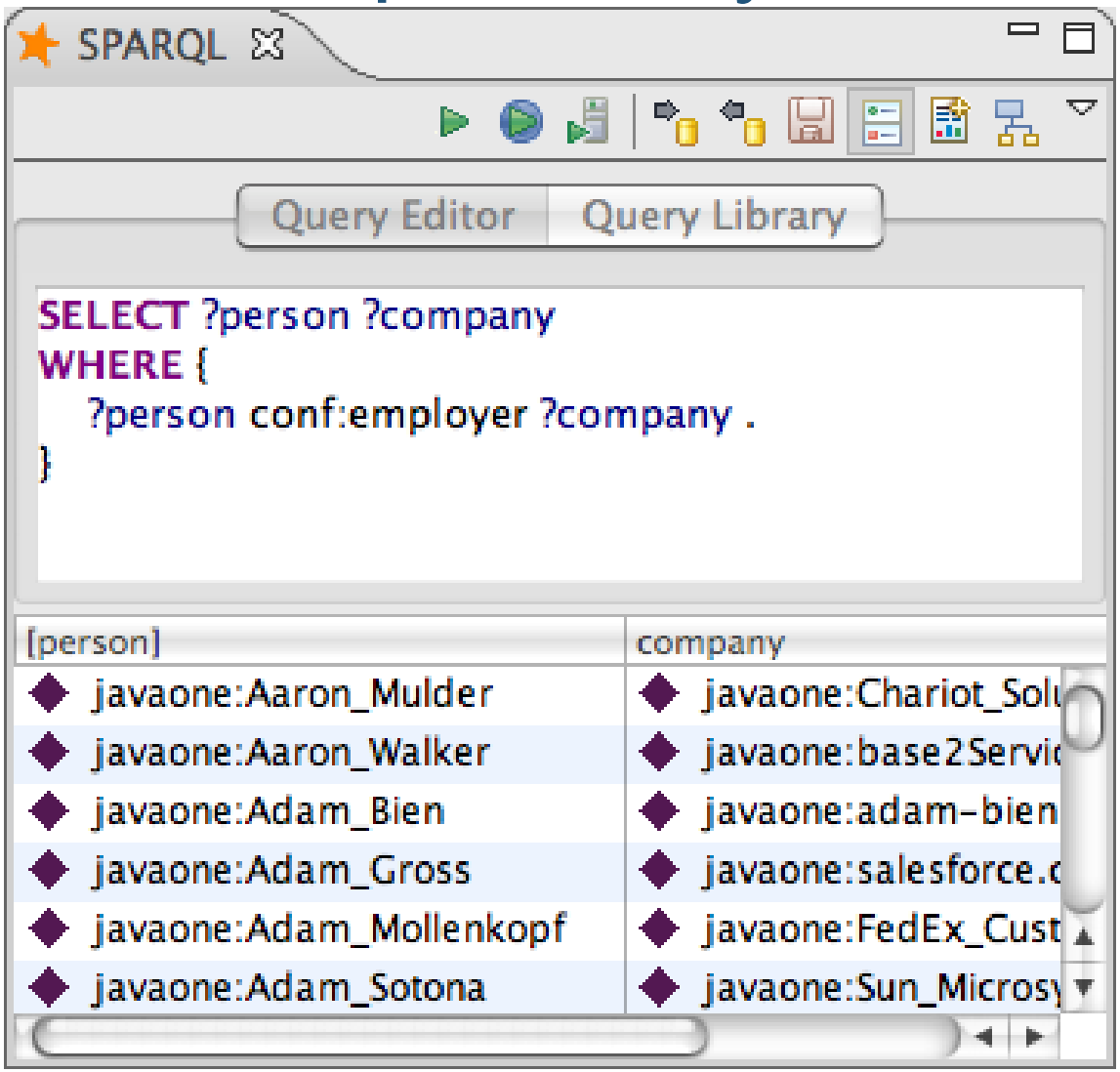
- RDF Reactor
  - http://semanticweb.org/wiki/RDFReactor
- ELMO
  - http://www.openrdf.org/doc/elmo/1.4/
- Jastor
  - http://jastor.sourceforge.net/
- OWL2Java

# Intermediate Summary

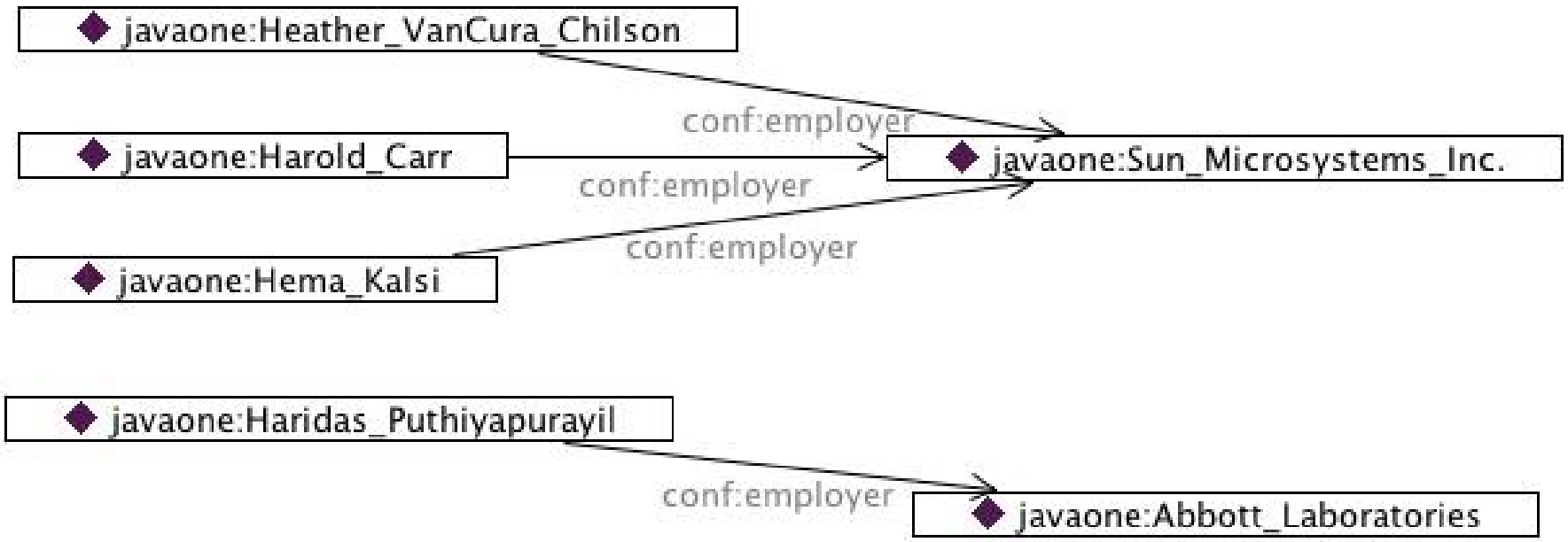
- Data model in RDF and RDF Schema
- Data binding to link RDF and Java worlds

- Next: let the data work for us with SPARQL

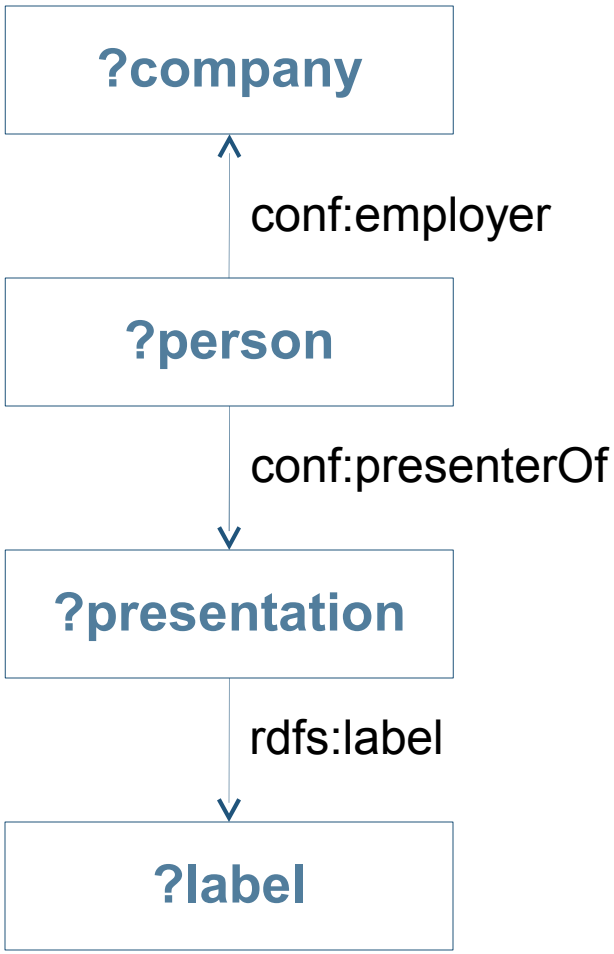
# SPARQL: Example Query



# SPARQL: Query results as subgraphs



# SPARQL: Example 2



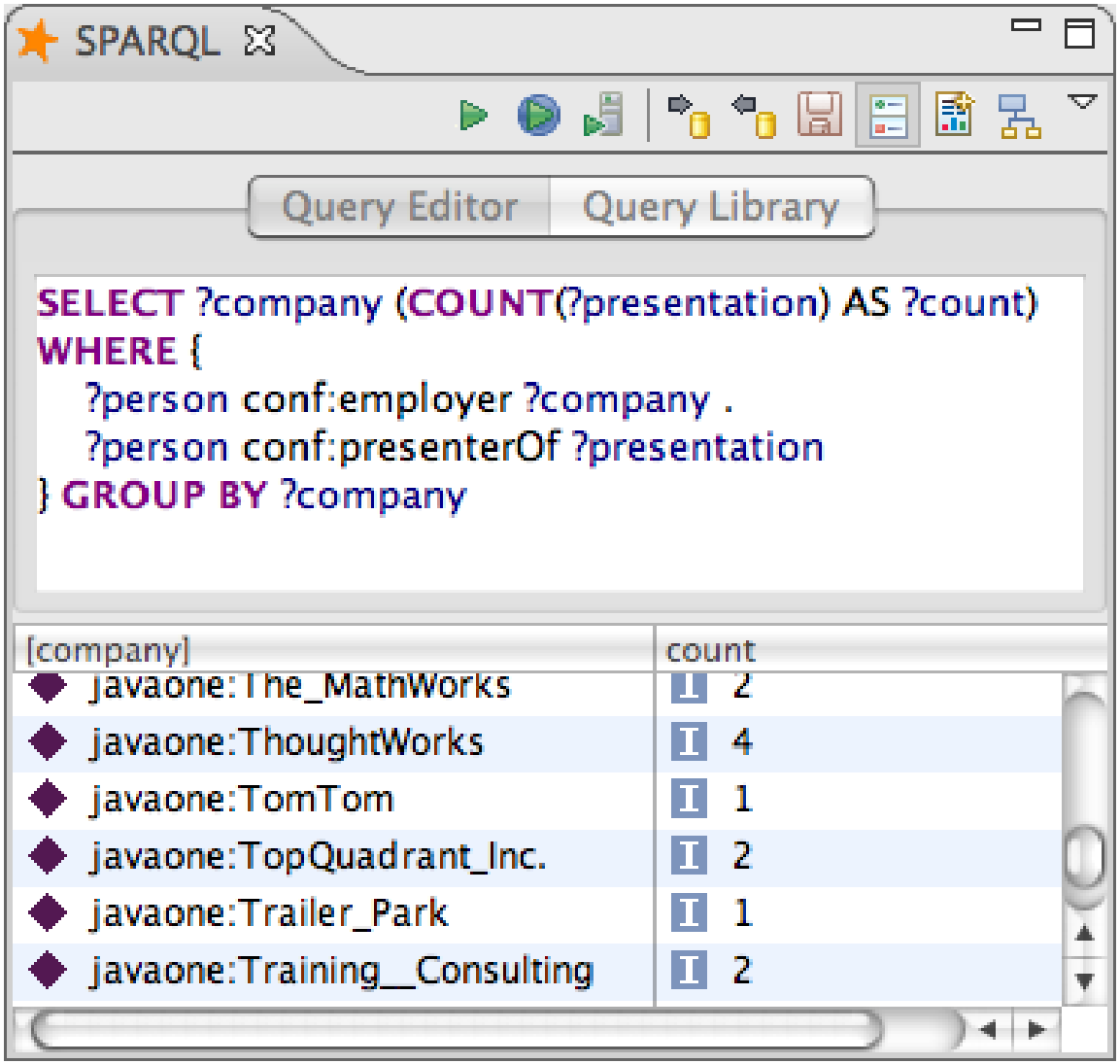

SPARQL

Query Editor | Query Library

```
SELECT ?company ?label
WHERE {
   ?person conf:employer ?company .
   ?person conf:presenterOf ?presentation .
   ?presentation rdfs:label ?label .
}
```

| [company] | label |
|---|---|
| javaone:Intel | Maximizing Javaa„¢ Tech |
| javaone:Intelligent_Softw | Griffon in Depth |
| javaone:Intuit | Cloud Computing: Show |
| javaone:Intuit | Web 2.0 Security Puzzle |
| javaone:JBoss | Bean Validation: Declare |
| javaone:JBoss | Full-Text Search: Humar |

# SPARQL: Example 3

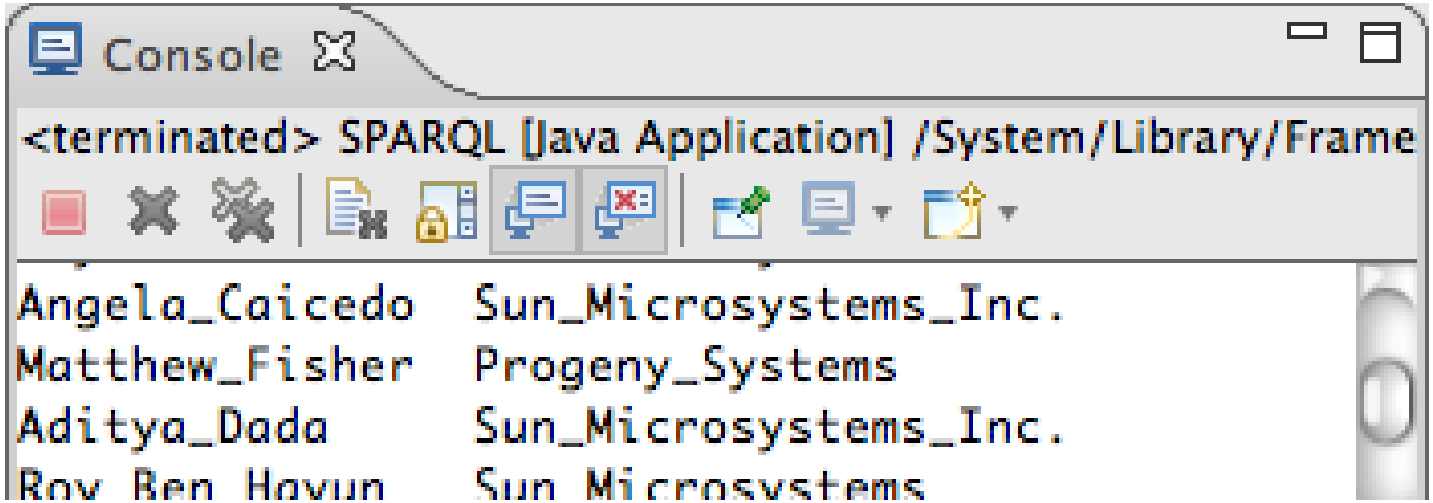


# Jena API: SPARQL

```
Model m = ModelFactory.createDefaultModel();
m.read(new FileReader("conf.n3"), null, FileUtils.langN3);
m.read(new FileReader("javaOne.rdf"), null);
String queryString =
    "PREFIX " + CONF.PREFIX + ": <" + CONF.NS + "> \n" +
    "SELECT ?person ?company \n" +
    "WHERE { \n" +
    "    ?person conf:employer ?company . \n" +
    "}";
Query query = QueryFactory.create(queryString);
QueryExecution qx = QueryExecutionFactory.create(query, m);
```

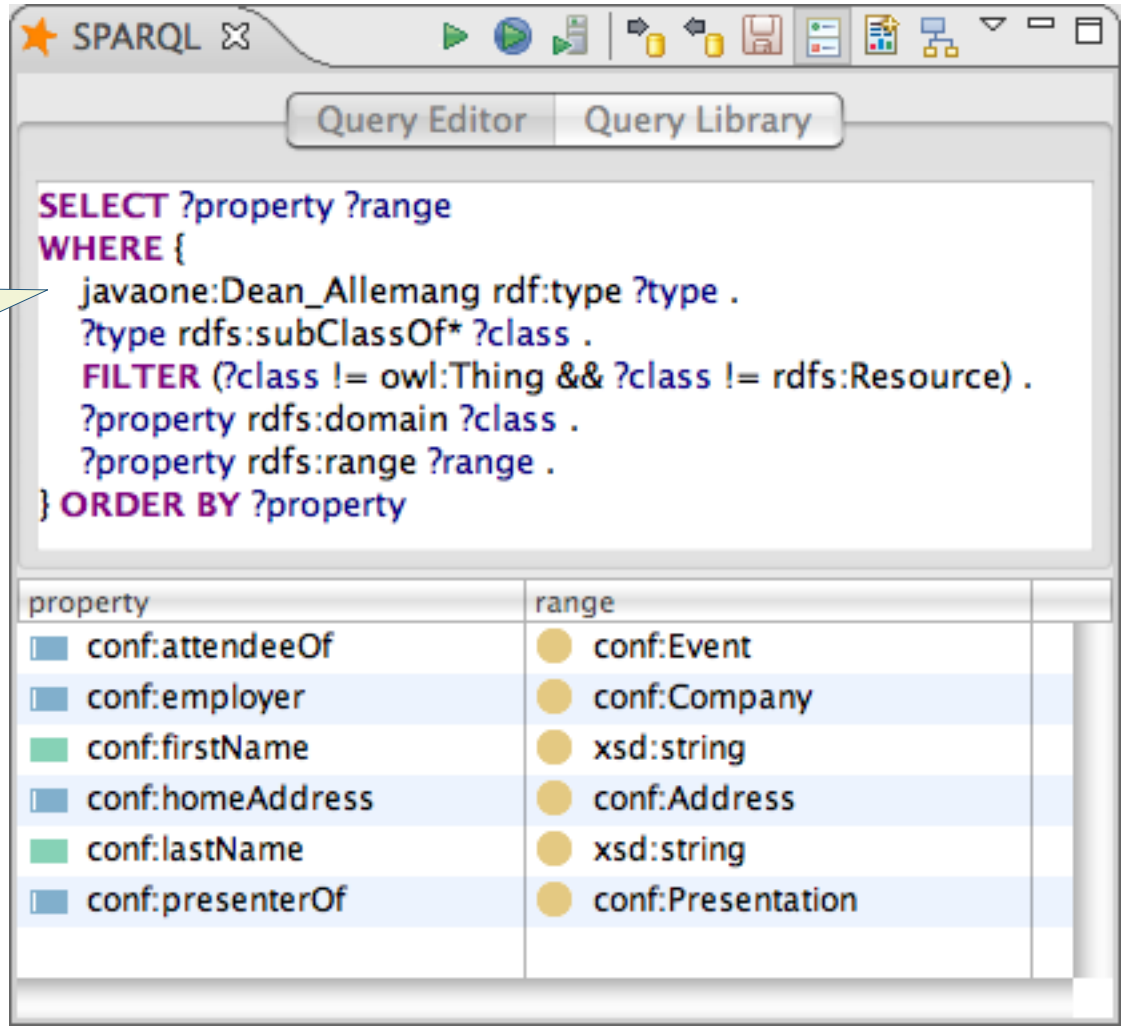
# Jena API: SPARQL (cont'd)

```
ResultSet rs = qx.execSelect();
while(rs.hasNext()) {
    QuerySolution s = rs.nextSolution();
    Resource person = (Resource) s.get("person");
    Resource company = (Resource) s.get("company");
    System.out.println(person.getLocalName() + "\t" +
                                company.getLocalName());
}
```

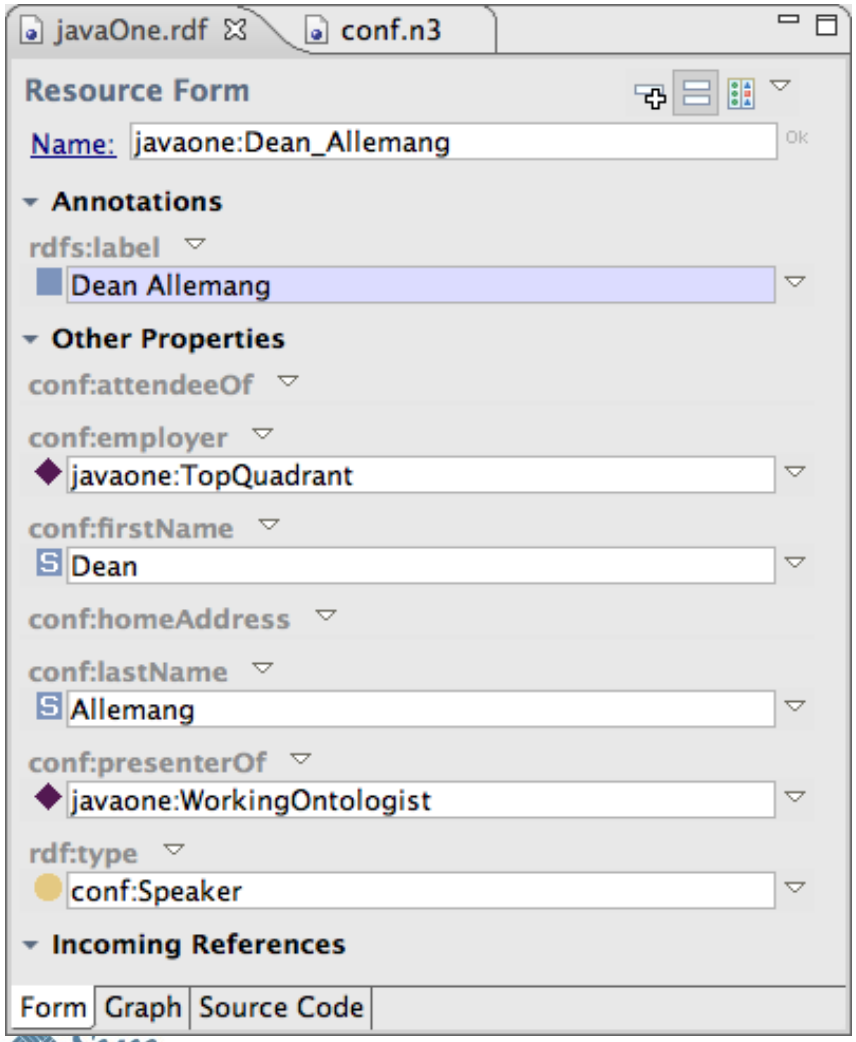


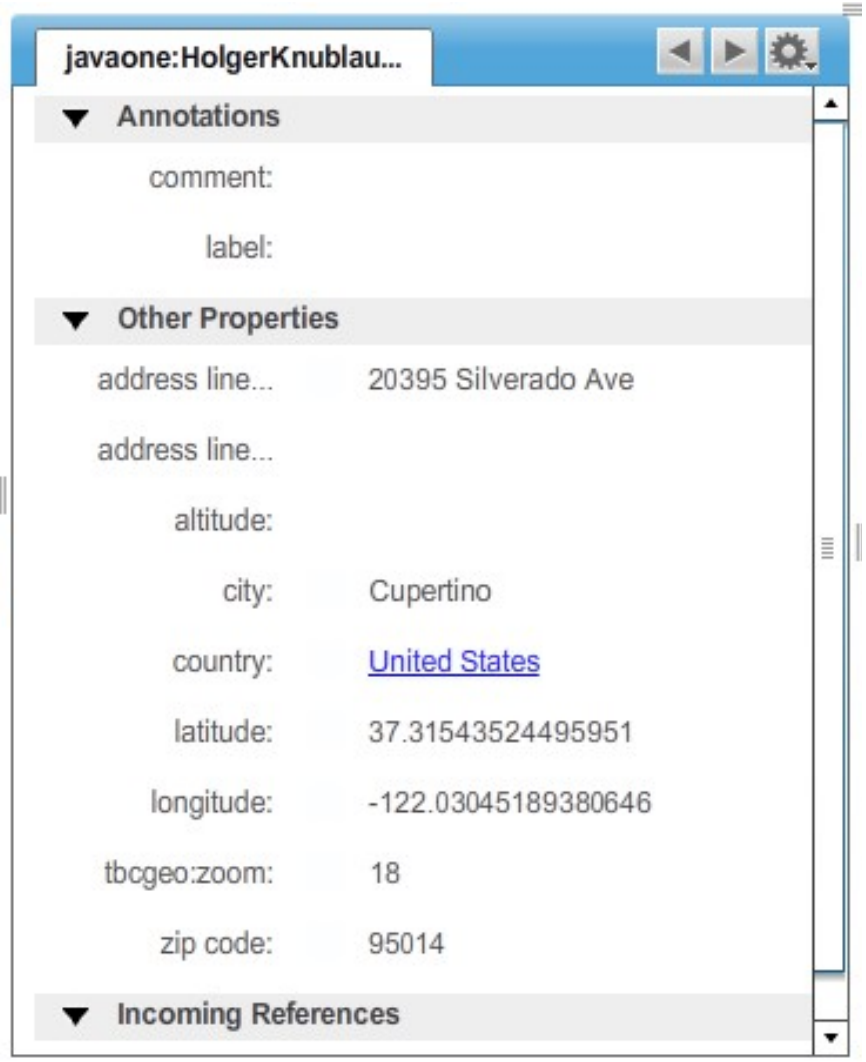
# SPARQL: Querying the schema

"Which properties are relevant for a given resource?"

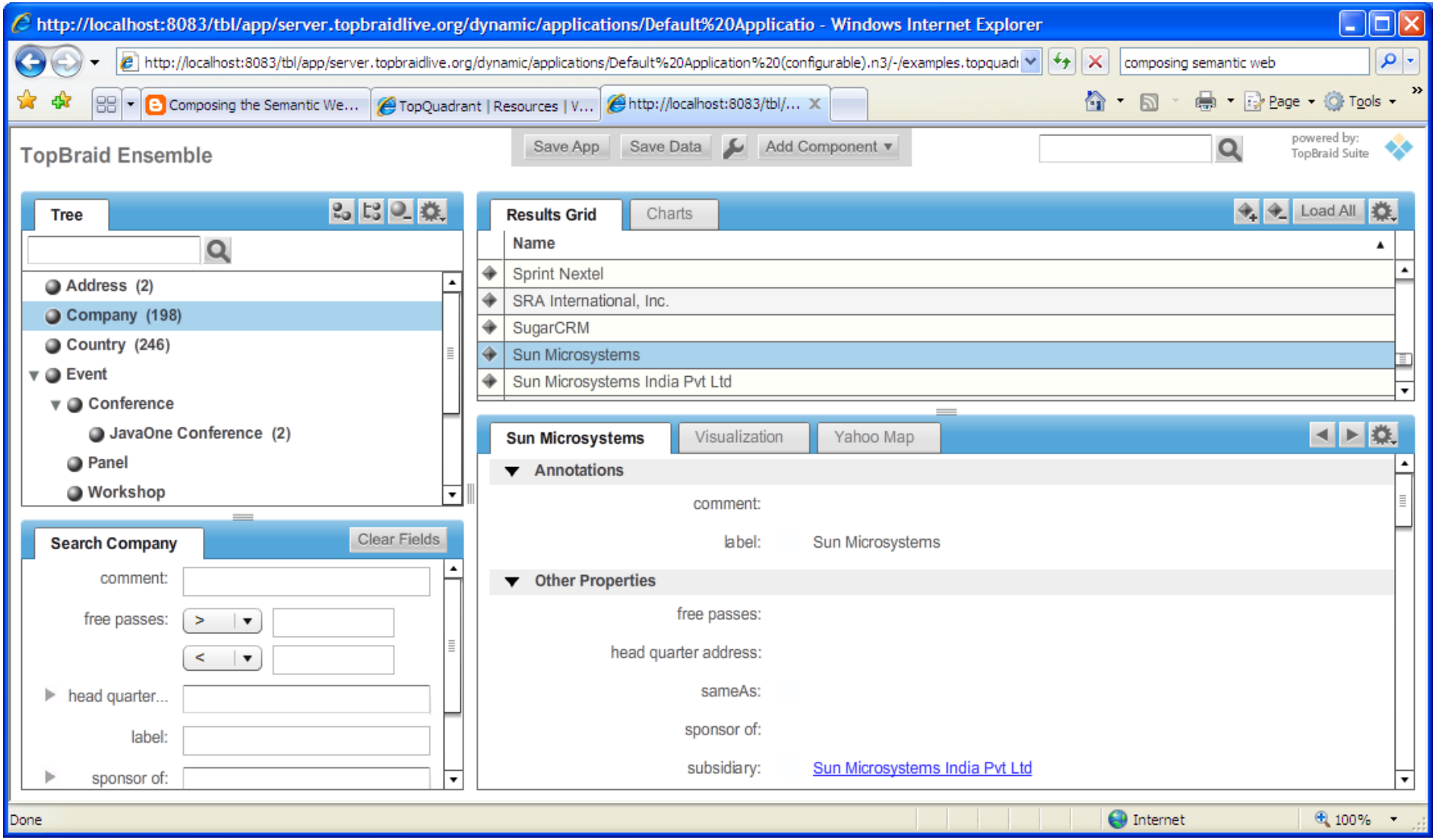


# SPARQL: Model-driven forms

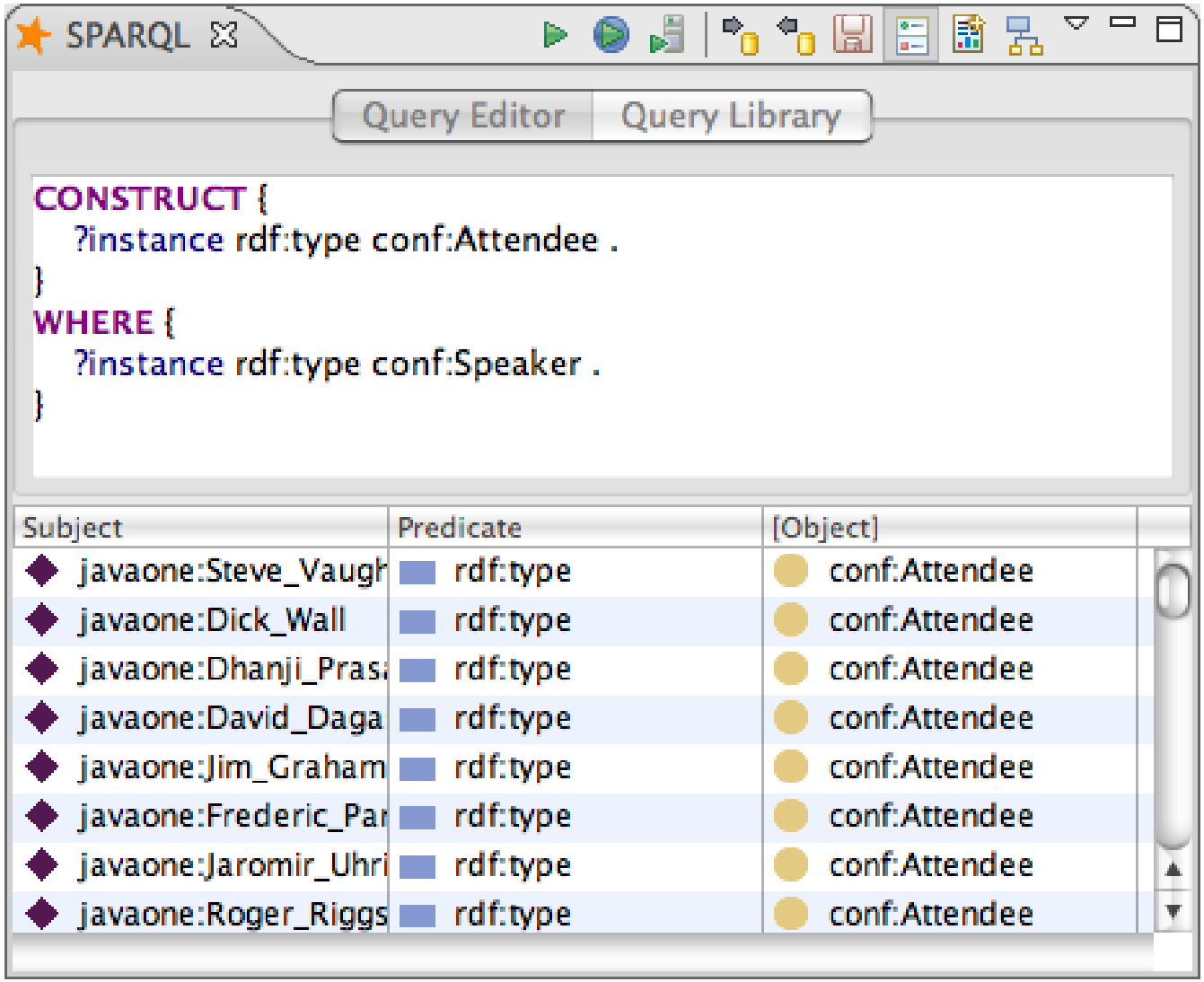


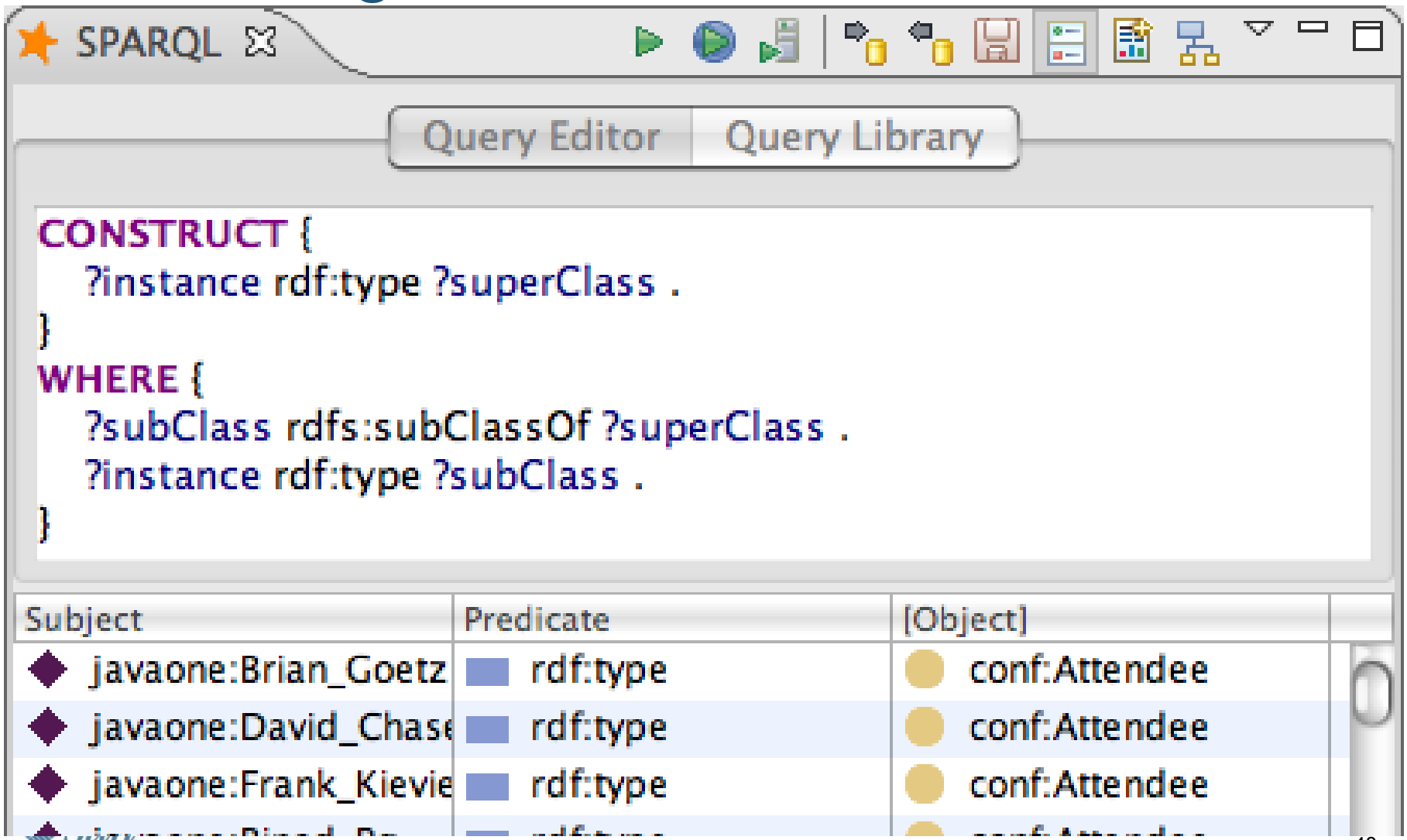
# SPARQL: Model-driven applications



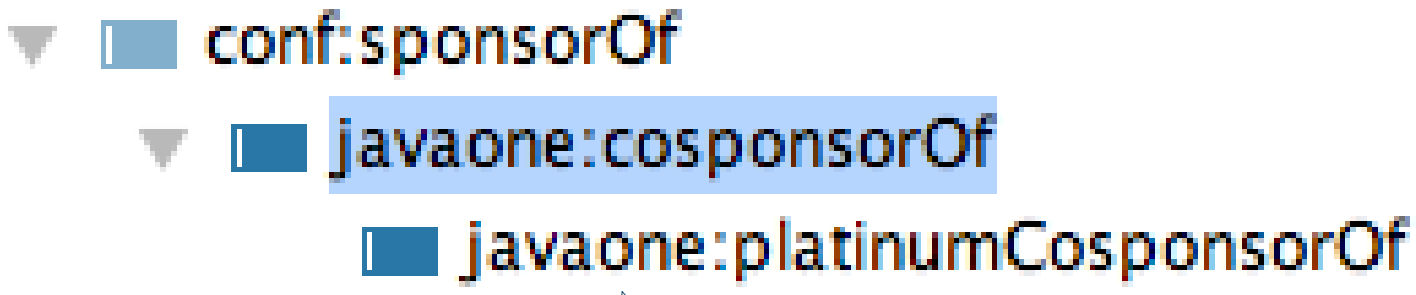
# SPARQL: CONSTRUCT rule



# SPARQL: generalized rule



# Rules: Extending the core schema



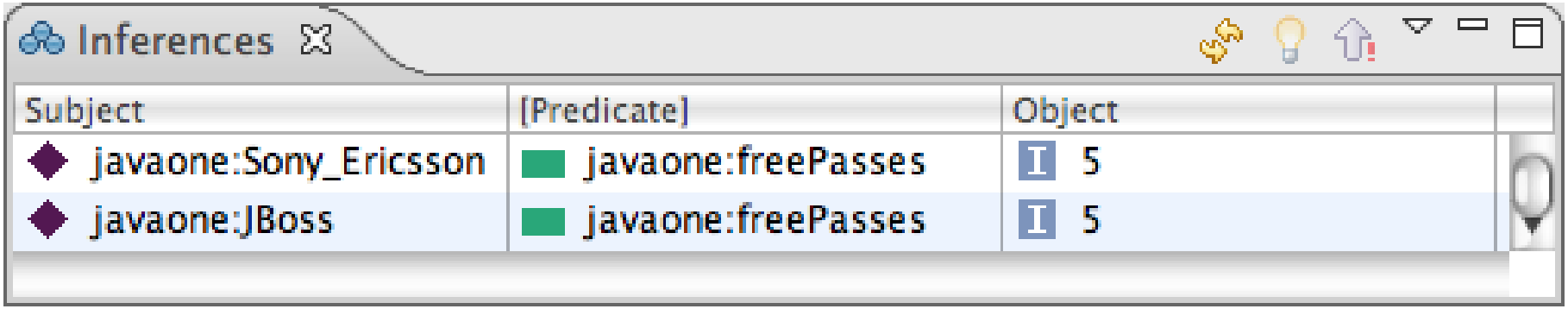

Adding domain-specific properties; the application's code knows nothing about them at compile-time.

# Rules: Conference Sponsorship

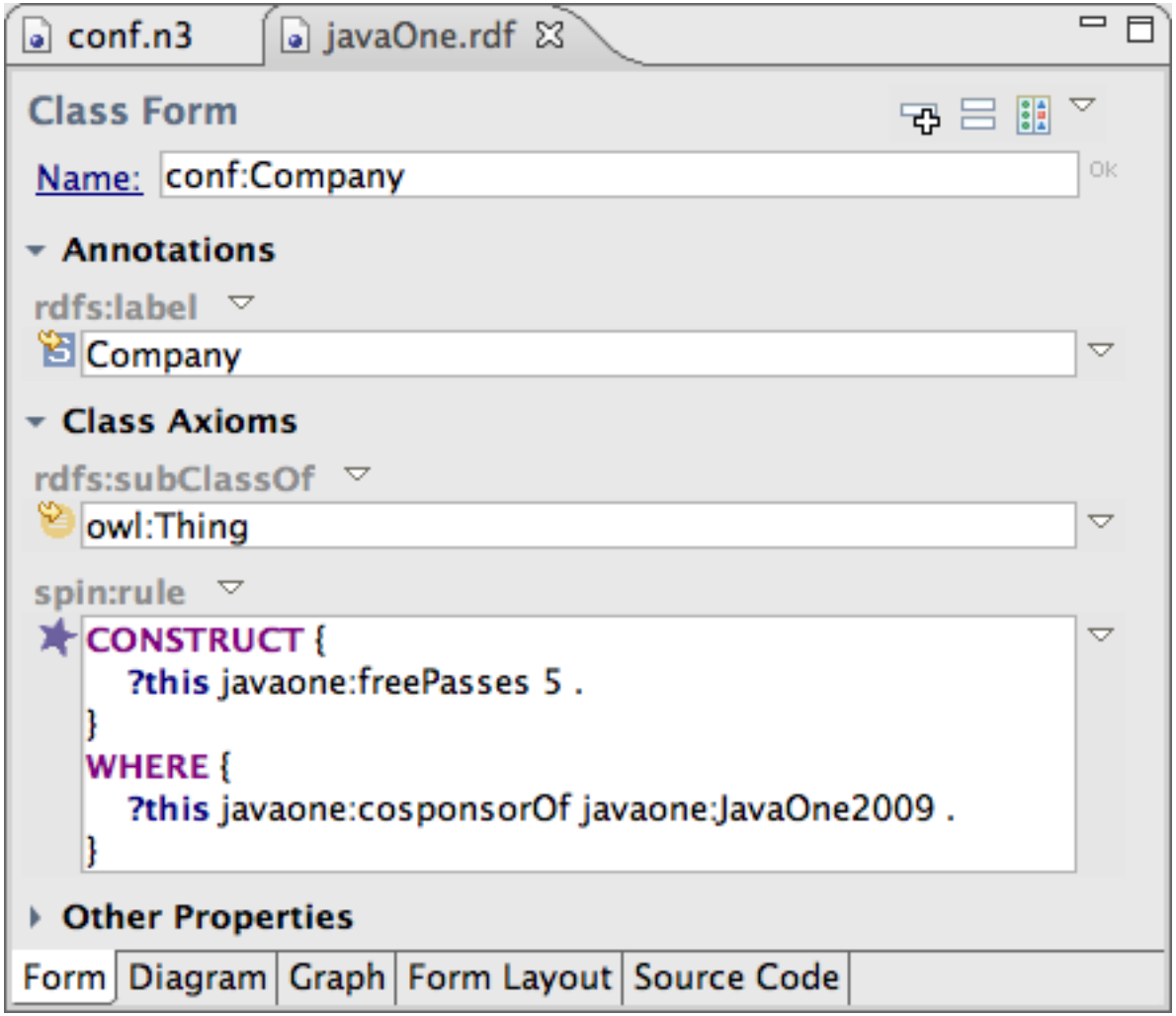
"Companies that are co-sponsors get 5 free conference passes"

Condition
(triple pattern in the graph)

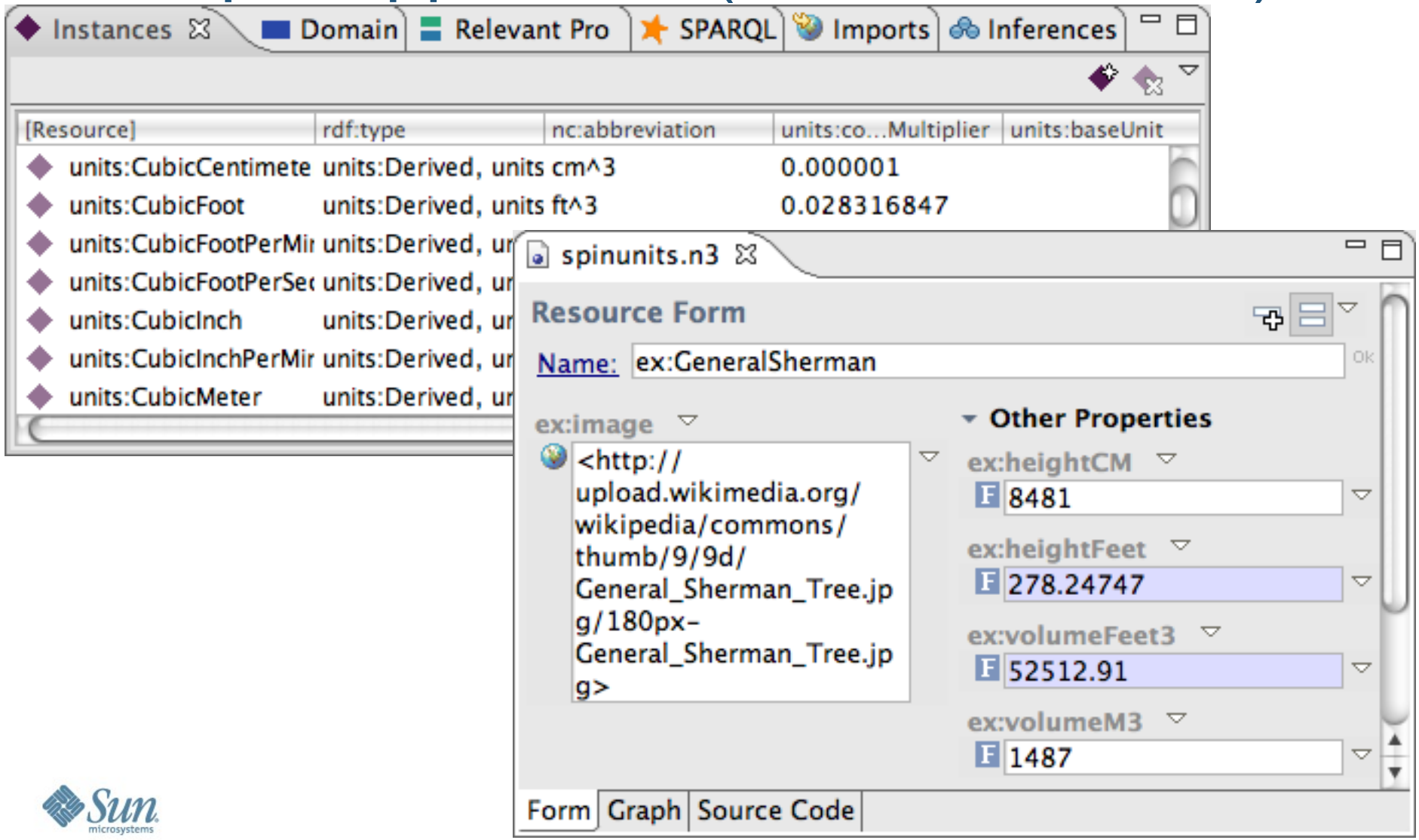
Action
(triples to add/infer)



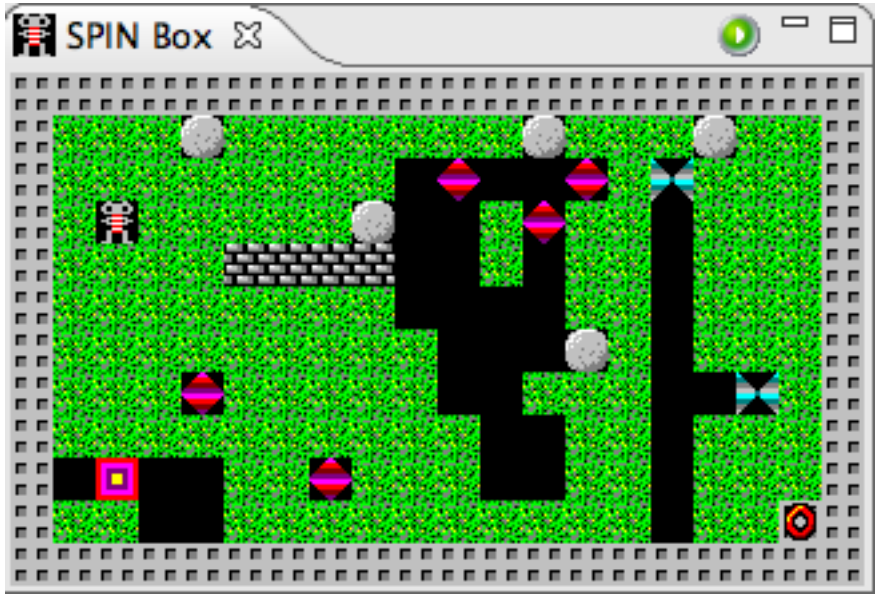
# Rules: SPARQL Inferencing Notation

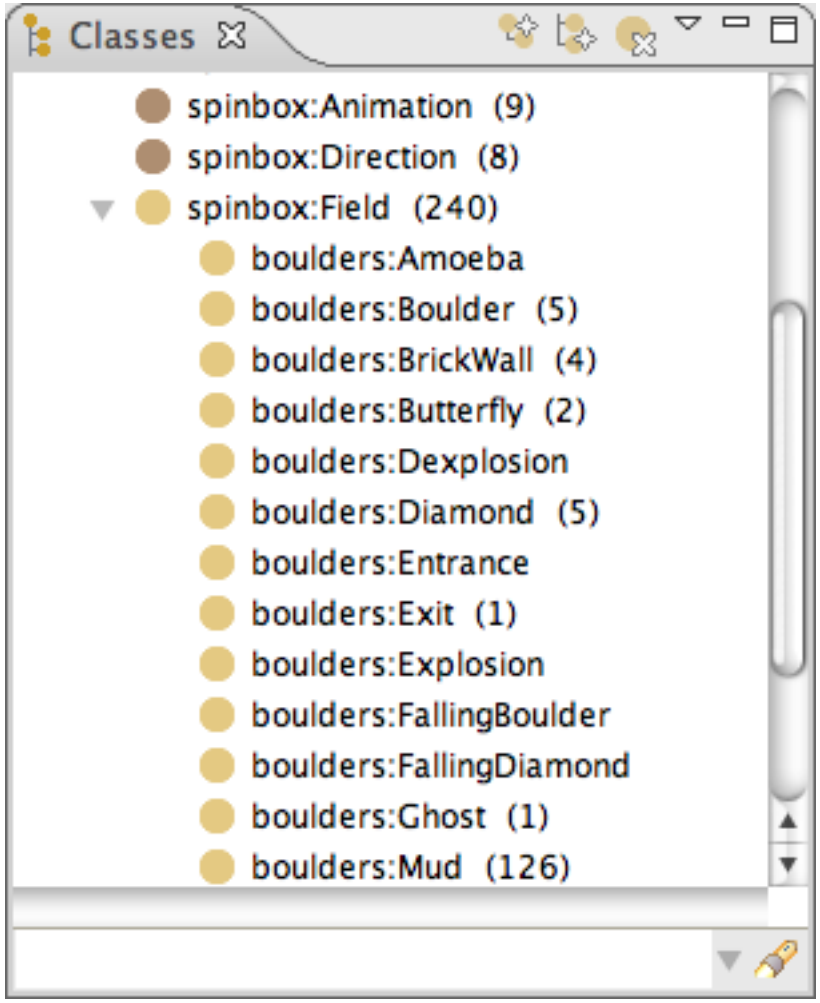


# Example Application (Unit Conversion)



# Example Application (Computer Game)




composing-the-semantic-web.blogspot.com

# Summary

- New development paradigm based on Linked Data
- Smart code walks the data graph at run-time
- Very little hard-coding of behavior
- Benefits
  - flexible architecture
  - faster turn-around times
- Challenges
  - learning curve (this is not OO!)
  - integration with established technologies


# Thank You

Holger Knublauch
holger@topquadrant.com

http://www.topquadrant.com

# Backup Slides

# RDF Databases

- AllegroGraph
- Jena (RDB, SDB, TDB)
- Mulgara
- Oracle 11g
- Sesame

# Inferencing APIs

- Often integrated into RDF stores:
  - Jena Rules
  - Sesame
  - Oracle 11g RDF
- OWLIM (Rule engine with RDFS/OWL support)
  - http://ontotext.com/owlim/
- Pellet (OWL DL)
  - http://clarkparsia.com/pellet/
- TopSPIN (Rules via SPARQL)
  - http://spinrdf.org/api