

# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

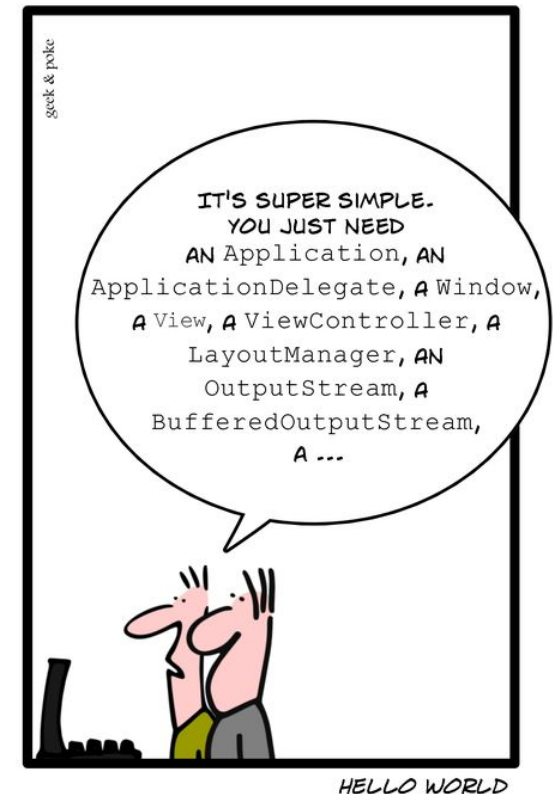
# Was bisher geschah

- Informationsrepräsentation mit RDF
- Schemata mit RDFS
- Bereitstellung der Information als Linked Data
- Ein wenig einfaches Reasoning
- Zugriff auf Information mit SPARQL
  
- Was uns noch fehlt
  - Wie kann man jetzt sinnvolle Anwendungen damit bauen?

# Programmierung von Semantic Web Applikationen

- Wie fangen wir da an?
- Eine einfache Hello-World-Anwendung...

*SIMPLY EXPLAINED*



<http://geekandpoke.typepad.com/geekandpoke/2010/06/hello-world.html>

# Programmierung mit (Java-)Bordmitteln



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
URL url = new URL("http://dbpedia.org/resource/Darmstadt");
URLConnection conn = url.openConnection();
conn.addRequestProperty("Accept", "text/rdf+n3");
BufferedReader BR = new BufferedReader(
    new InputStreamReader(conn.getInputStream())
);

while (BR.ready()) {
    String triple = BR.readLine();
    StringTokenizer tokenizer = new StringTokenizer(triple, " ");
    String subject = tokenizer.nextToken();
    String predicate = tokenizer.nextToken();
    String object = tokenizer.nextToken();
    ...
}
```

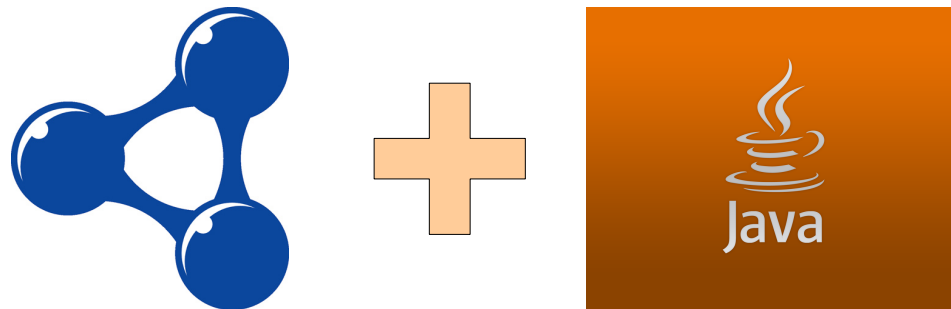


# Programmierung mit Frameworks

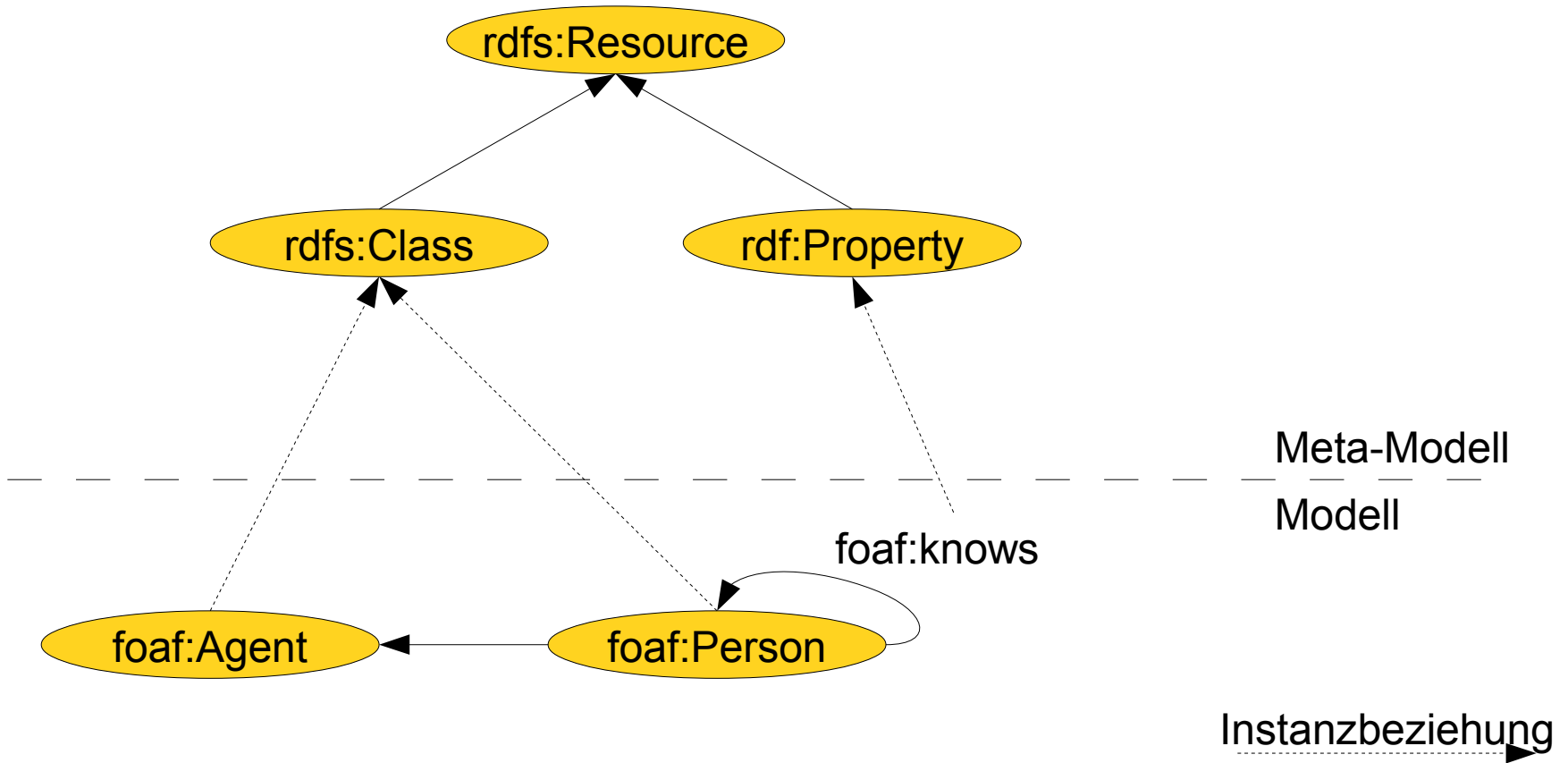


TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Mit Bordmitteln ist das möglich, aber aufwändig
- Besser: spezialisierte Frameworks verwenden
- Wir lernen jetzt einige kennen



# Direkte und indirekte Programmier-Frameworks



# Direkte und indirekte Programmier-Frameworks



- Direkte Programmierframeworks:

- Java-Klassen entsprechen Klassen im RDF-Schema
- z.B. für FOAF:

```
Person p = new Person();  
Person.setName("Peter");
```

- Indirekte Programmierframeworks:

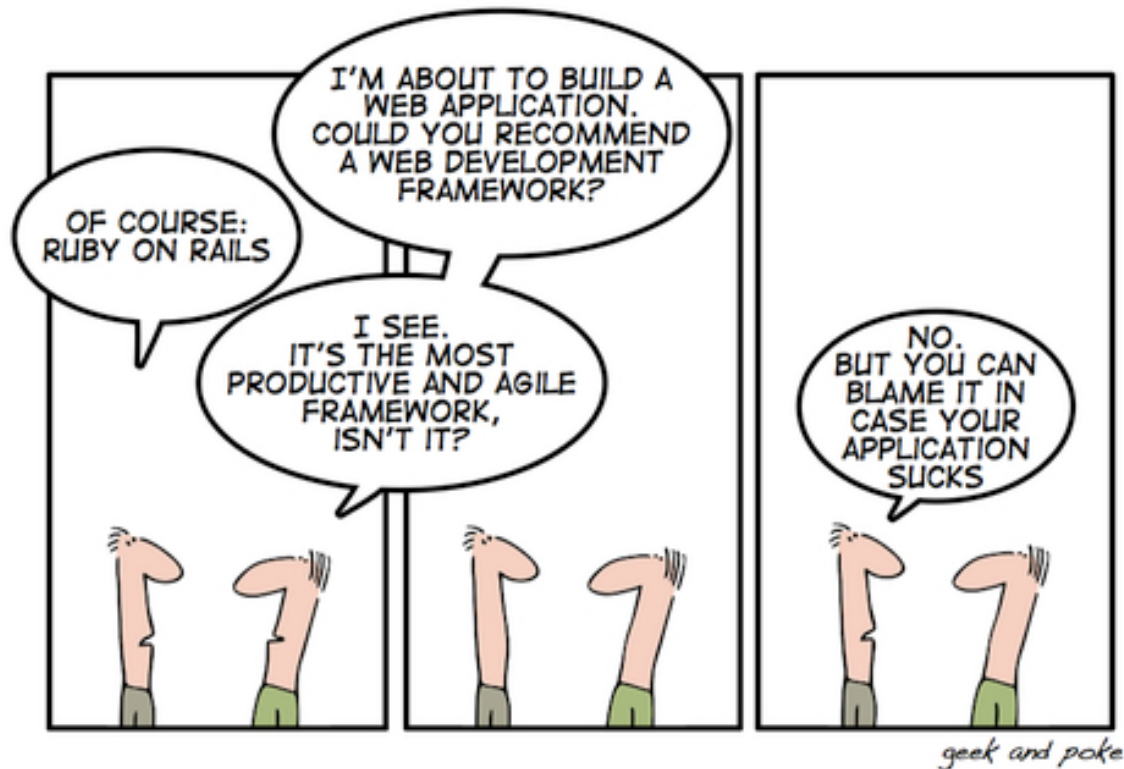
- Java-Klassen entsprechen Klassen im RDF-Metamodell
- z.B. für FOAF:

```
RDFNode p = new RDFNode("foaf:Person");  
p.setAttributeValue("foaf:name", "Peter");
```

# Direkte und indirekte Programmierframeworks



- Man muss sich hier entscheiden



<http://geekandpoke.typepad.com/geekandpoke/2008/05/how-to-choose-a.html>

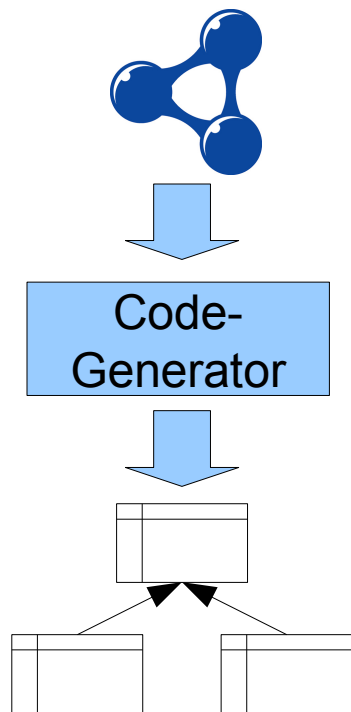


# Direkte und indirekte Programmier-Frameworks



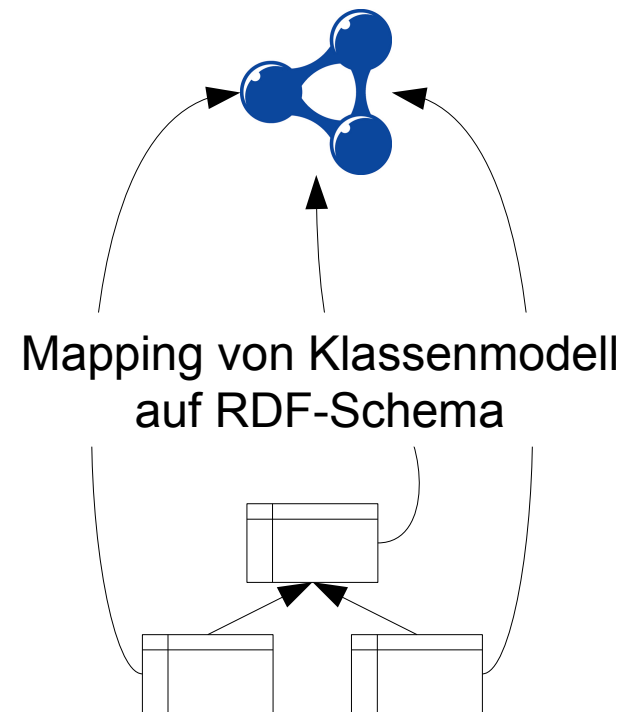
- Direkte Programmier-Frameworks
  - Gut lesbarer Code
  - Intuitiv
  - aber: nicht auf unbekanntem Schemata anwendbar
  - und: das Schema bestimmt die Programmierung!
- Indirekte Programmier-Frameworks
  - weniger intuitiv
  - mehr Overhead
  - flexibler: Schema muss nicht bekannt sein
  - flexibler: das Schema kann sich weiterentwickeln

- Mögliche Ausprägungen:
  - Generierung von Klassenmodell aus Schema
  - Binden von existierenden Klassenmodellen an ein Schema



RDF-Schema

Klassenmodell



# Beispiel: RDFReactor



- RDFReactor: Generiert Klassen aus Schema
- Diese können dann zur Verarbeitung von RDF genutzt werden
- Die Basis ist ein RDF2Go-Modell:

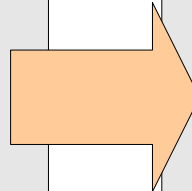


```
Model model = RDF2Go.getModelFactory().createModel();
```

# Beispiel: RDFReactor und FOAF

- Erzeugen eines Klassenmodells aus FOAF
  - mit dem Code-Generator von RDFReactor

```
foaf:Person a rdfs:Class .
foaf:knows a rdf:Property.
foaf:knows
  rdfs:domain foaf:Person ;
  rdfs:range foaf:Person .
...
```



```
public class Person {
  private Person[] knows;

  public void addKnows(Person p) {
    ...
  }

  public Person[] getAllKnows() {
    return knows;
  }

  ...
}
```

# Nutzung der generierten Klassen



- Beispiel: RDF von einer URI lesen und verarbeiten

```
model.read(new InputStreamReader(urlConn.getInputStream()));

Iterator<Person> persons =
    Person.getAllInstances_as(model);

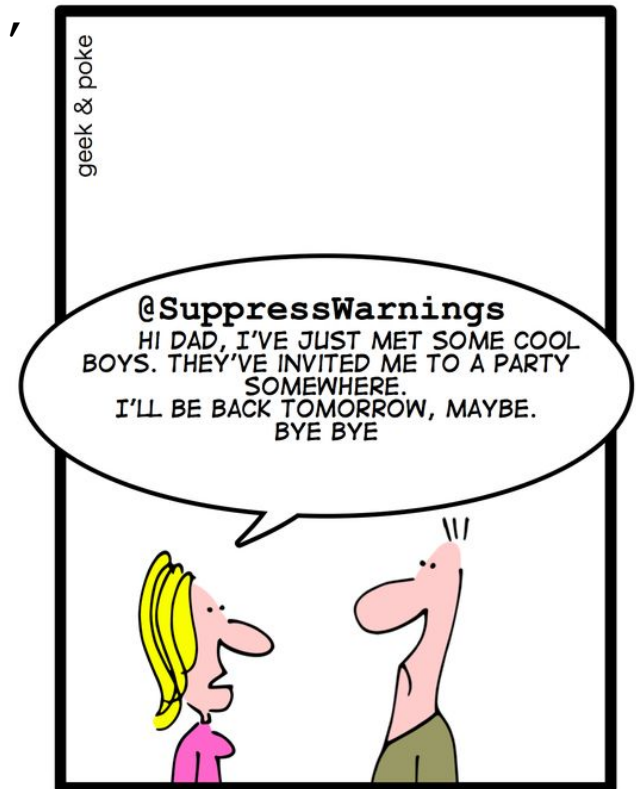
while (persons.hasNext() {
    Person[] friends = persons.next().getAllKnows();
    ...
}
```

# Nutzung bestehender Klassen: Java Annotations

## ▪ Beispiel: otm-j

```
@RDF(base="http://xmlns.com/foaf/0.1/",  
      uri="Person")  
interface Person extends RDFResource {  
    @RDF("name")  
    String getName();  
}
```

*SIMPLY EXPLAINED*



<https://projects.quasthoffs.de/otm-j>  
<http://geekandpoke.typepad.com/geekandpoke/2010/03/simply-explained-annotations.html>

# Nutzung bestehender Klassen: Java Annotations

- Beispiel: otm-j

```
QueryObject query = graph.query(Person.class)
                        .where("name", "Peter");

for(Person p : query.select()) {
    System.out.println(p.getMbox());
}
```

<https://projects.quasthoffs.de/otm-j>

# Nicht-intrusive Varianten

- Generierung von Code:
  - nur praktisch bei neuen Anwendungen
- Nutzung von Annotations
  - wenn Klassen im Source-Code vorliegen
  - und modifiziert werden können/dürfen
- Nicht-intrusive Varianten
  - Auslagern von Mappings in eigene Datei
  - z.B. ELMO: <http://www.openrdf.org/doc/elmo/1.5/>



# Direkte Programmier-Frameworks



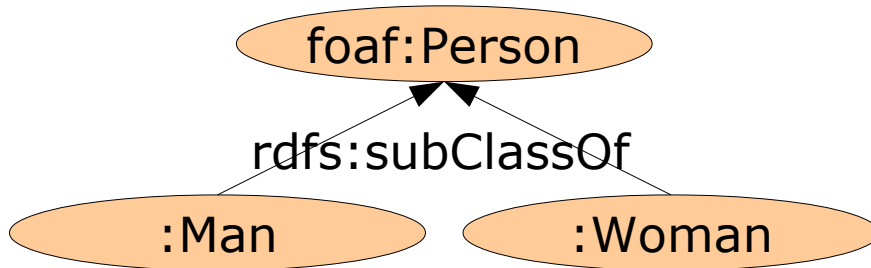
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Häufiges Problem:
  - Ähnlichkeit zwischen RDF-Schema und Klassenmodell
    - angenommen (1:1-Abbildung)
    - aber nicht gegeben
- Mögliche Lösung:
  - Nutzung von Abbildungsregeln

# Regelbasierte Abbildung RDF-Schema ↔ Klassenmodell



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



```
public class Person {  
    private enum Gender {male, female};  
    private Gender gender;  
  
    public Gender getGender() {  
        return gender;  
    }  
  
    ...  
}
```

# Regelbasierte Abbildung RDF-Schema ↔ Klassenmodell



```
@RDF (base="http://xmlns.com/foaf/0.1/",  
      uri="Person")
```

- Ungenaue Abbildung
- Verlustfreies Serialisieren/  
Deserialisieren nicht möglich
- Was wir bräuchten:
  - dynamische Abbildung
  - wenn gender=male,  
dann :Man, sonst :Woman

```
public class Person {  
    private enum Gender {male, female};  
    private Gender gender;  
  
    public Gender getGender() {  
        return gender;  
    }  
  
    ...  
}
```

# Regelbasierte Abbildung RDF-Schema ↔ Klassenmodell



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Java nach RDF:

Person[Gender=male] → . rdf:type :Man .

- RDF nach Java

{?p a :Man} → createObject(Person).setValue(gender,"male").

Paulheim et al.: "Mapping Pragmatic Class Models to Reference Ontologies".  
2<sup>nd</sup> Workshop on Data Engineering Meets the Semantic Web, 2011.



# Indirekte Programmier-Frameworks



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Direkte Programmier-Frameworks sind praktisch
  - wenn man vorher weiß, was man will
  - domänenspezifische Anwendungen
    - Adressbuch
    - Büchereiverwaltung
    - ...
- Für bestimmte Szenarien passt das aber nicht
  - z.B. Entwicklung eines allgemeinen Semantic-Web-Browsers



# Beispiel: Jena



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Jena ist ein Standard-Framework für Semantic-Web-Anwendungen
- seit 2010 von HP Labs entwickelt
  
- Zentrale Konzepte
  - Modelle (`class Model`)
  - Ressourcen (`class Resource`)
- Besonderheiten
  - Datenbankbindung für Persistenz
  - Unterstützung von SPARQL
  - Reasoning, Nutzung von Ontologien
  - Regel-Verarbeitung



# Beispiel: Jena

- Modell lesen von einer URL

```
model.read("http://dbpedia.org/resource/Darmstadt");
```

- Durch Modelle navigieren

```
Resource darmstadt =  
    model.getResource("http://dbpedia.org/resource/  
                    Darmstadt");
```

```
Resource federalState =  
    darmstadt.getProperty(  
        "http://dbpedia.org/ontology/federalState").  
    getResource();
```

# Beispiel: Jena

- Arbeiten mit Literalen

```
Literal lit = darmstadt.getProperty(  
    "http://www.w3.org/2000/01/rdf-schema#label") .  
    getLiteral();  
  
lit.getString();  
lit.getLanguage();  
lit.getDatatype();
```



# Beispiel: Jena

## ▪ Arbeiten mit mehrwertigen Relationen

```
StmtIterator iter = darmstadt.getProperty(
"http://www.w3.org/2000/01/rdf-schema#label");

while(iter.hasNext()) {
    Statement s = iter.next();
    RDFNode node = s.getObject();
    if(node.isLiteral())
        System.out.println(node.asLiteral().getString());
}
}
```

erzeugt einen Iterator über alle  
Tripel mit dem Subjekt-Knoten  
und dem angegebenen Prädikat

# Beispiel: Jena

- Manipulation von Modellen

```
p1.addProperty("http://xmlns.com/foaf/0.1/knows", p2);
```

- Modelle überwachen

```
class MyListener implements ModelChangeListener...  
MyListener listener = new MyListener();  
model.add(listener);
```

# Beispiel: Jena + Reasoning

- Wir erinnern uns: aus Schema (T-Box) und Daten (A-Box) können wir Information ableiten
  - :knows rdfs:domain :Person .
  - :knows rdfs:range :Person .
  - :Peter :knows :Tom .
  - :Peter a :Person . :Tom a :Person .
- Das kann Jena auch

# Beispiel: Jena + Reasoning

- Gegeben: ein Schema und eine Datensammlung

```
Model schemaModel = ModelFactory.createDefaultModel();  
InputStream IS = new FileInputStream("data/example_schema.rdf");  
schemaModel.read(IS);
```

```
Model dataModel = ModelFactory.createDefaultModel();  
IS = new FileInputStream("data/example_data.rdf");  
dataModel.read(IS);
```

```
Model reasoningModel =  
    ModelFactory.createRDFSModel(schemaModel, dataModel);
```

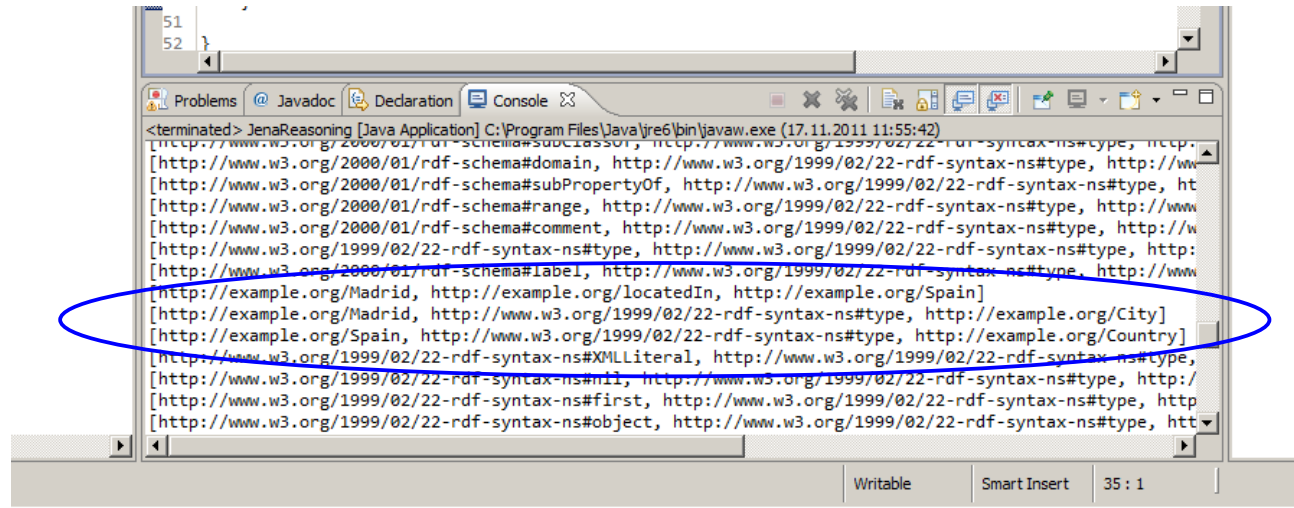
- Das `reasoningModel` enthält jetzt alle abgeleiteten Fakten!

# Beispiel: Jena + Reasoning

- Das reasoningModel enthält jetzt alle abgeleiteten Fakten!

```
StmtIterator it = reasoningModel.listStatements();  
while(it.hasNext()) {  
    Statement s = it.next();  
    System.out.println(s);  
}
```

- Ausgabe:



```
<terminated> JenaReasoning [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (17.11.2011 11:55:42)  
[http://www.w3.org/2000/01/rdf-schema#subClassOf, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://ww  
[http://www.w3.org/2000/01/rdf-schema#domain, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://ww  
[http://www.w3.org/2000/01/rdf-schema#subPropertyOf, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, ht  
[http://www.w3.org/2000/01/rdf-schema#range, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://www  
[http://www.w3.org/2000/01/rdf-schema#comment, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://w  
[http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http:  
[http://www.w3.org/2000/01/rdf-schema#label, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://www  
[http://example.org/Madrid, http://example.org/locatedIn, http://example.org/Spain]  
[http://example.org/Madrid, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://example.org/City]  
[http://example.org/Spain, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://example.org/Country]  
[http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral, http://www.w3.org/1999/02/22-rdf-syntax-ns#type,  
[http://www.w3.org/1999/02/22-rdf-syntax-ns#nil, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://  
[http://www.w3.org/1999/02/22-rdf-syntax-ns#first, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http:  
[http://www.w3.org/1999/02/22-rdf-syntax-ns#object, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, htt
```

# Beispiel: Jena + Reasoning

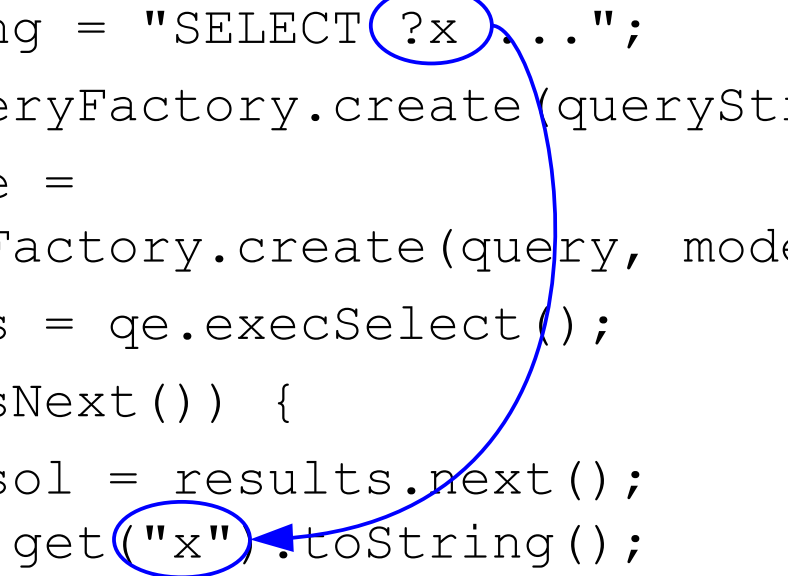


- Jena kommt mit mehreren eingebauten Reasonern
  - RDFS
  - OWL
  - Regeln
- Manchmal möchte man aber andere benutzen
  - Performance
  - Abdeckung
- Lösung: externer Reasoner, z.B. Pellet:
  - ```
OntModel model = ModelFactory.createOntologyModel(  
    PelletReasonerFactory.THE_SPEC );
```

# Beispiel: Jena + SPARQL

- Modell mit SPARQL "befragen"

```
String queryString = "SELECT ?x ...";
Query query = QueryFactory.create(queryString);
QueryExecution qe =
    QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();
while(results.hasNext()) {
    QuerySolution sol = results.next();
    String s = sol.get("x").toString();
    ...
}
```



# Beispiel: Jena + SPARQL + Reasoning

- Mit SPARQL greift man auch auf alle abgeleiteten Fakten zu
- Gegeben wieder unser `reasoningModel`

```
Query q = QueryFactory.create(
    "SELECT ?t WHERE
    { <http://example.org/Madrid>
      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        ?t .}" );
QueryExecution qexec =
    QueryExecutionFactory.create(q, reasoningModel);
ResultSet rs = qexec.execSelect();
while(rs.hasNext())
    String type = rs.next().get("t");
```

- Damit findet man zwei Lösungen:
  - `http://example.org/City`
  - `http://www.w3.org/2000/01/rdf-schema#Resource`



# Zugriff auf öffentliche SPARQL-Endpoints



- Wir erinnern uns:
  - SPARQL-Endpoints sind ein Baustein im Semantic Web
- Zugriff mit Jena:

```
String query = "SELECT ...";  
String endpoint = "http://dbpedia.org/sparql";  
Query q = QueryFactory.create(strQuery);  
QueryExecution qexec =  
    QueryExecutionFactory.sparqlService(endpoint, q);  
ResultSet RS = qexec.executeSelect();
```

# Hybride Frameworks



- Wir haben gesehen
  - direkte Frameworks sind bequem
  - indirekte Frameworks sind flexibel
- Vorschlag von Puleston et al. (2008):
  - hybrides Framework
  - direktes Framework für häufig genutzte Top-Konzepte
  - indirektes Framework für Sub-Konzepte

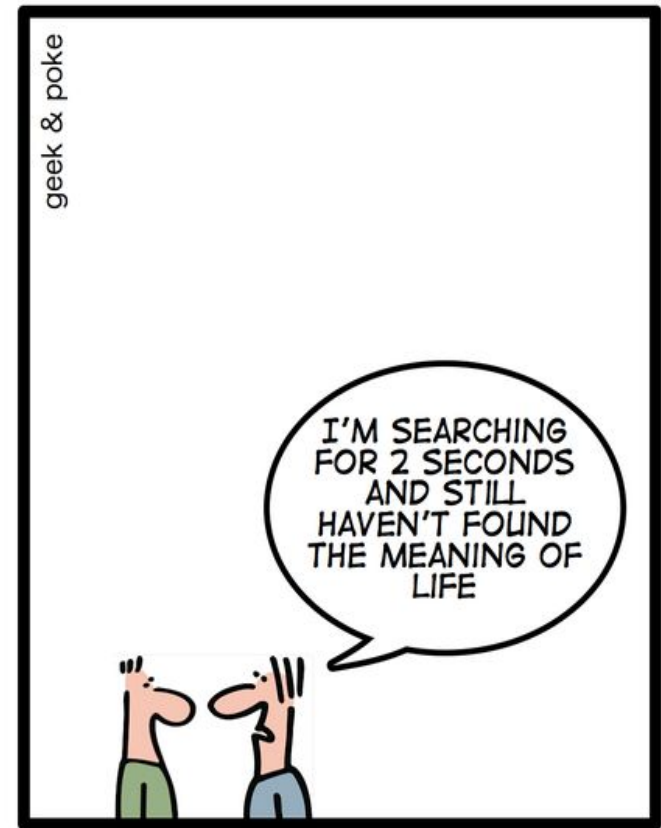
# Weitere Tools und Bibliotheken



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Was können wir bis jetzt?
  - RDF verarbeiten, lesen, speichern
  - Abfragen
  - Reasoning
  
- Was gibt es sonst noch?
  - Suchen
  - Taggen
  - Visualisieren
  - ...

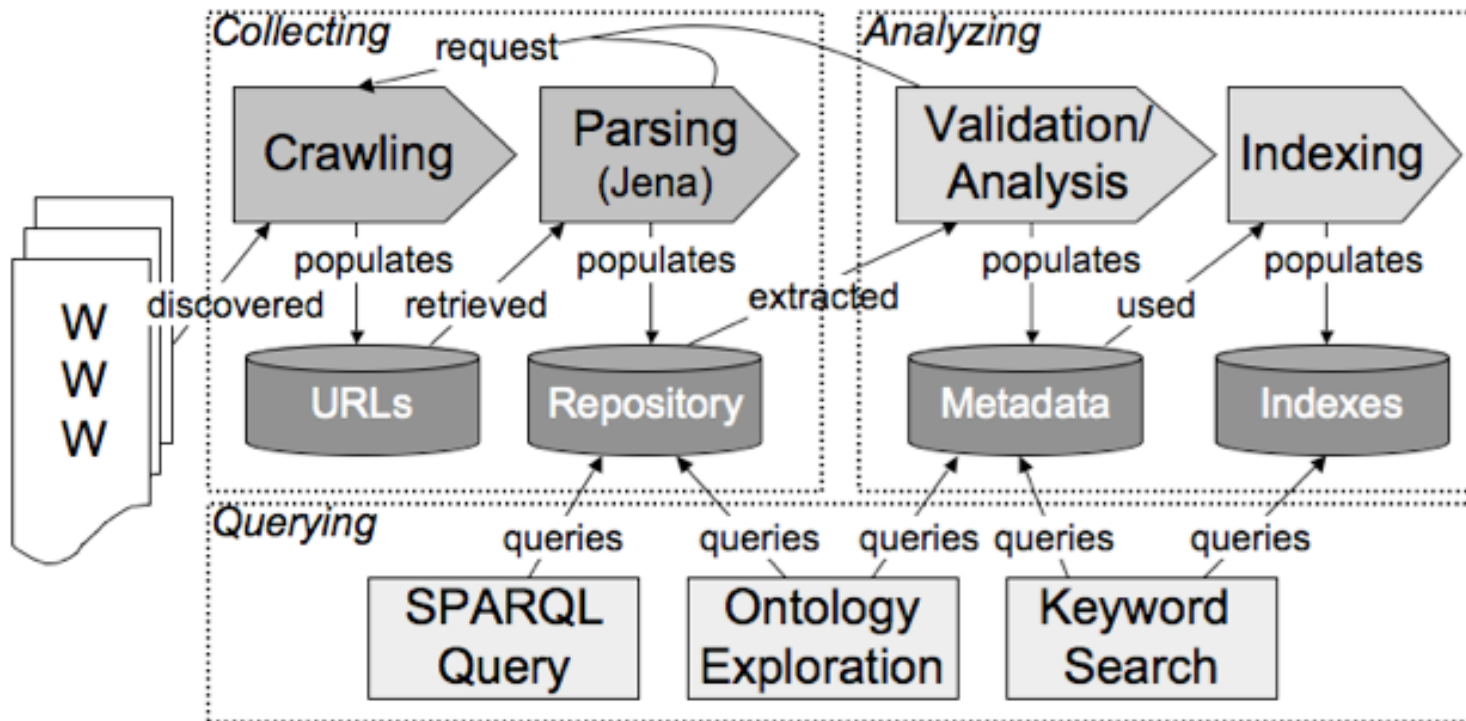
- Wie haben wir bis jetzt im Semantic Web gesucht?
  - URIs "raten"
  - über SPARQL mit Labels



*INSTANT SEARCH*

<http://geekandpoke.typepad.com/geekandpoke/2010/09/instant-search>.


- Beispiel: RDF Watson – sucht RDF-Dokumente



<http://watson.kmi.open.ac.uk/Overview.html>

# Suchen

- RDF-Watson
  - SOAP-Schnittstelle
  - Java-API



The screenshot shows a Firefox browser window with the URL `watson.kmi.open.ac.uk/WatsonWUI/`. The search results for 'Darmstadt' are displayed, showing 30 semantic documents. The first 10 results are listed below:

- 1- [http://users.livejournal.com/\\_sixpence\\_/data/foaf/](http://users.livejournal.com/_sixpence_/data/foaf/)  
♦ [http://www.livejournal.com/directory.bml?opt\\_sort=ut&s\\_loc=1&loc\\_cn=DE&loc\\_st=&loc\\_ci=Darmstadt](http://www.livejournal.com/directory.bml?opt_sort=ut&s_loc=1&loc_cn=DE&loc_st=&loc_ci=Darmstadt)
- 2- <http://www.talkdigger.com/sioc/www.itst.net/xml/>  
♦ <http://blog.eigenfrequenz.net/2007/05/14/ajax-vortrag-an-der-tu-darmstadt>  
♦ <http://www.local-pages.de/forum/viewtopic.php?p=1478>
- 3- [http://swordfish.rdfweb.org/photos/rdf/000783\\_rdf/](http://swordfish.rdfweb.org/photos/rdf/000783_rdf/)  
♦ <http://swordfish.rdfweb.org/photos/2001/09/07/000783.JPG>
- 4- [http://swordfish.rdfweb.org/photos/rdf/000782\\_rdf/](http://swordfish.rdfweb.org/photos/rdf/000782_rdf/)  
♦ <http://swordfish.rdfweb.org/photos/2001/09/07/000782.JPG>
- 5- <http://delimiter.livejournal.com/data/foaf/>  
♦ [http://www.livejournal.com/directory.bml?opt\\_sort=ut&s\\_loc=1&loc\\_cn=DE&loc\\_st=&loc\\_ci=Darmstadt](http://www.livejournal.com/directory.bml?opt_sort=ut&s_loc=1&loc_cn=DE&loc_st=&loc_ci=Darmstadt)
- 6- [http://swordfish.rdfweb.org/photos/rdf/000785\\_rdf/](http://swordfish.rdfweb.org/photos/rdf/000785_rdf/)  
♦ <http://swordfish.rdfweb.org/photos/2001/09/07/000785.JPG>
- 7- <http://zutroy.livejournal.com/data/foaf/>  
♦ [http://www.livejournal.com/directory.bml?opt\\_sort=ut&s\\_loc=1&loc\\_cn=DE&loc\\_st=&loc\\_ci=Darmstadt](http://www.livejournal.com/directory.bml?opt_sort=ut&s_loc=1&loc_cn=DE&loc_st=&loc_ci=Darmstadt)
- 8- [http://mirrors.webthing.com/view=Medium-Links/www.w3.org/People/Connolly/events/events-smart\\_rdf/](http://mirrors.webthing.com/view=Medium-Links/www.w3.org/People/Connolly/events/events-smart_rdf/)  
♦ [http://kmi-web05.open.ac.uk:81/cache/0/a24/e764/337b1/0750a051ae/3516978f17b3c7d63#place\\_Germany\\_Darmstadt](http://kmi-web05.open.ac.uk:81/cache/0/a24/e764/337b1/0750a051ae/3516978f17b3c7d63#place_Germany_Darmstadt)
- 9- [http://www.foodloos.de/50128-Jagdschloss\\_Kranichstein\\_rdf/](http://www.foodloos.de/50128-Jagdschloss_Kranichstein_rdf/)  
♦ <http://www.hotel-jagdschloss-kranichstein.de/>
- 10- <http://reliant.teknowledge.com/DAML/Elements.owl/>

# Suchen

- Sindice
  - Semantic-Web-Index
  - Suche mit Stichworten und Patterns
  - Ergebnisse in JSON, RDF/XML
  - Java-API verfügbar



Firefox

semantic web programmi... Category:Tool - Semanti... Category:Search Engine ... Watson - Semantic Web ... Category:Search Engine ... Search the Data Web... x

www.sindice.com/search?q=Darmstadt

Twiki Aigaion xkcd UL6 IEEE Xplore - Home SW Forum Lesezeichen

indice THE SEMANTIC WEB INDEX

About Search Submit Your Data Jobs Support Dev

Search interface type: [Simple](#) [Advanced](#) [Guru](#) [Query Language](#) [Documentation](#)

keyword(s)

SEARCH Group By Dataset:  Sorted by: [relevance](#)

Quick filters ([All options](#))

Time range:  
[Any date](#)  
[Today](#) [Yesterday](#) [Last week](#)  
[Last month](#) [Last year](#)

Format:  
[Any format](#)  
[RDF](#) [RDFa](#) [Microdata](#)  
[Microformat](#) [XFN](#) [HCard](#)  
[HCalendar](#) [HListing](#)  
[HResume](#) [License](#) [Geo](#)  
[ADR](#)

Predicate: [?](#)

Class: [?](#)

Sindice search: **Darmstadt** found 25,611 documents (in 0.03 seconds)

[Darmstadt](#) (RDF)  
2009-09-22 – 15 in 2.3 kB  
<http://dbpedia.org/resource/Category:Darmstadt> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

[Darmstadt](#) (RDF)  
2010-11-24 – 25 in 3.4 kB  
<http://semanticweb.org/id/Darmstadt> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

["Darmstadt"](#) (RDF)  
2011-09-23 – 7 in 756 Bytes  
<http://www.boardturk.com/ansiklopedi/114764-darmstadt.html> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

["Biergarten Darmstadt - Darmstadt - Biergärten"](#) (RDF)  
2011-09-22 – 38 in 4.9 kB  
<http://www.qype.com/place/31890-Biergarten-Darmstadt-Darmstadt> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

["Hauptbahnhof Darmstadt - Darmstadt - Bahnhöfe"](#) (RDF)  
2011-09-22 – 34 in 4.4 kB  
<http://www.qype.com/place/27946-Hauptbahnhof-Darmstadt-Darmstadt> ([Search](#)) [Inspect](#): ([Cache](#)) ([Live](#))

x Suchen:  [Abwärts](#) [Aufwärts](#) [Hervorheben](#)  Groß-/Kleinschreibung [Das Seitenende wurde erreicht, Suche vom Seitenanfang fortgesetzt](#)

# Suchen: Sindice

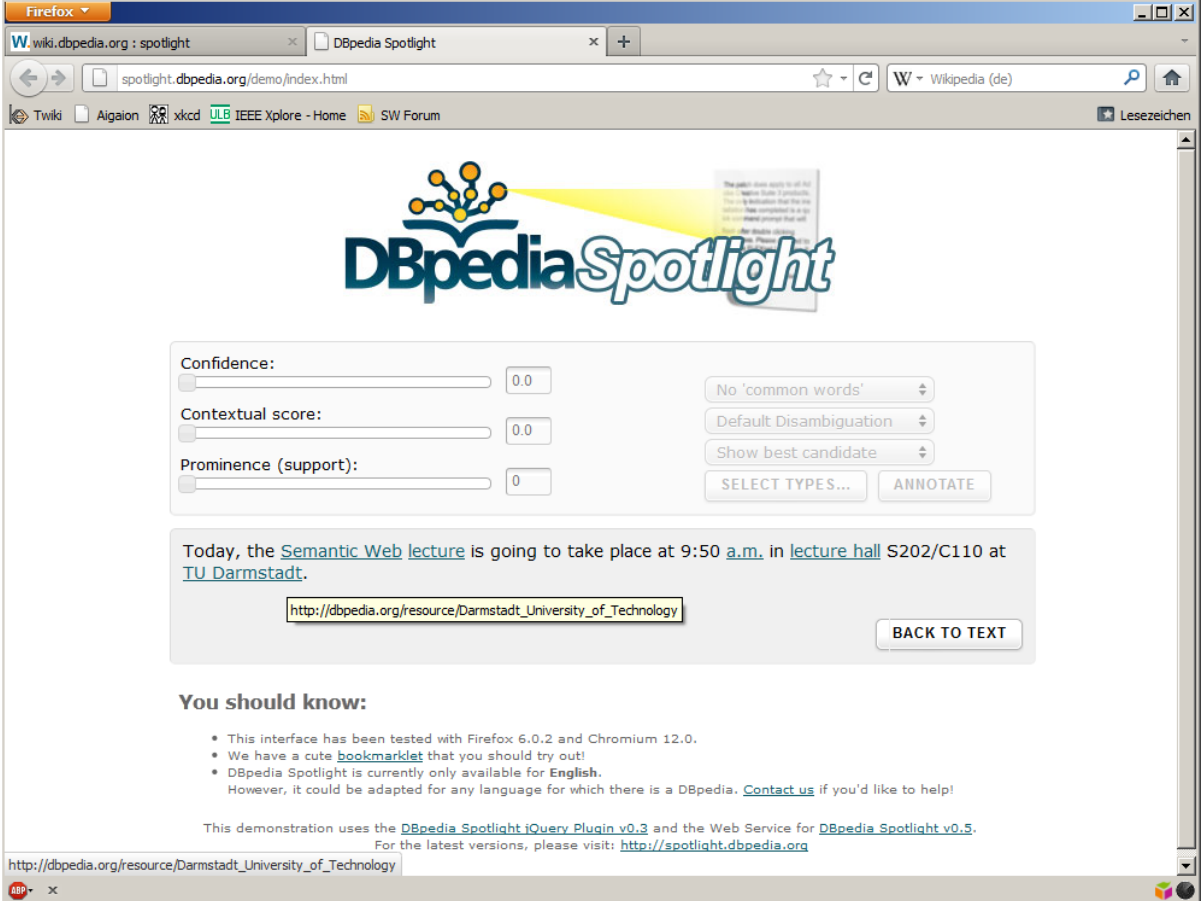


- Suchen mit Keywords
  - z.B. "TU Darmstadt", "Tim Berners-Lee"
  - ähnlich Google: AND, OR, NOT, exakter String
- Suchen mit Patterns (= Tripel mit Platzhalter)
  - z.B. \* foaf:name \*, \* dbpedia:location dbpedia:Germany
  - auch hier: AND, OR, NOT
- Beides kann kombiniert werden



# Tagging

- Text mit Links ins Semantic Web anreichern



Confidence:

Contextual score:

Prominence (support):

No 'common words'

Default Disambiguation

Show best candidate

SELECT TYPES... ANNOTATE

Today, the [Semantic Web lecture](#) is going to take place at 9:50 a.m. in [lecture hall S202/C110](#) at [TU Darmstadt](#).

[http://dbpedia.org/resource/Darmstadt\\_University\\_of\\_Technology](http://dbpedia.org/resource/Darmstadt_University_of_Technology) [BACK TO TEXT](#)

**You should know:**

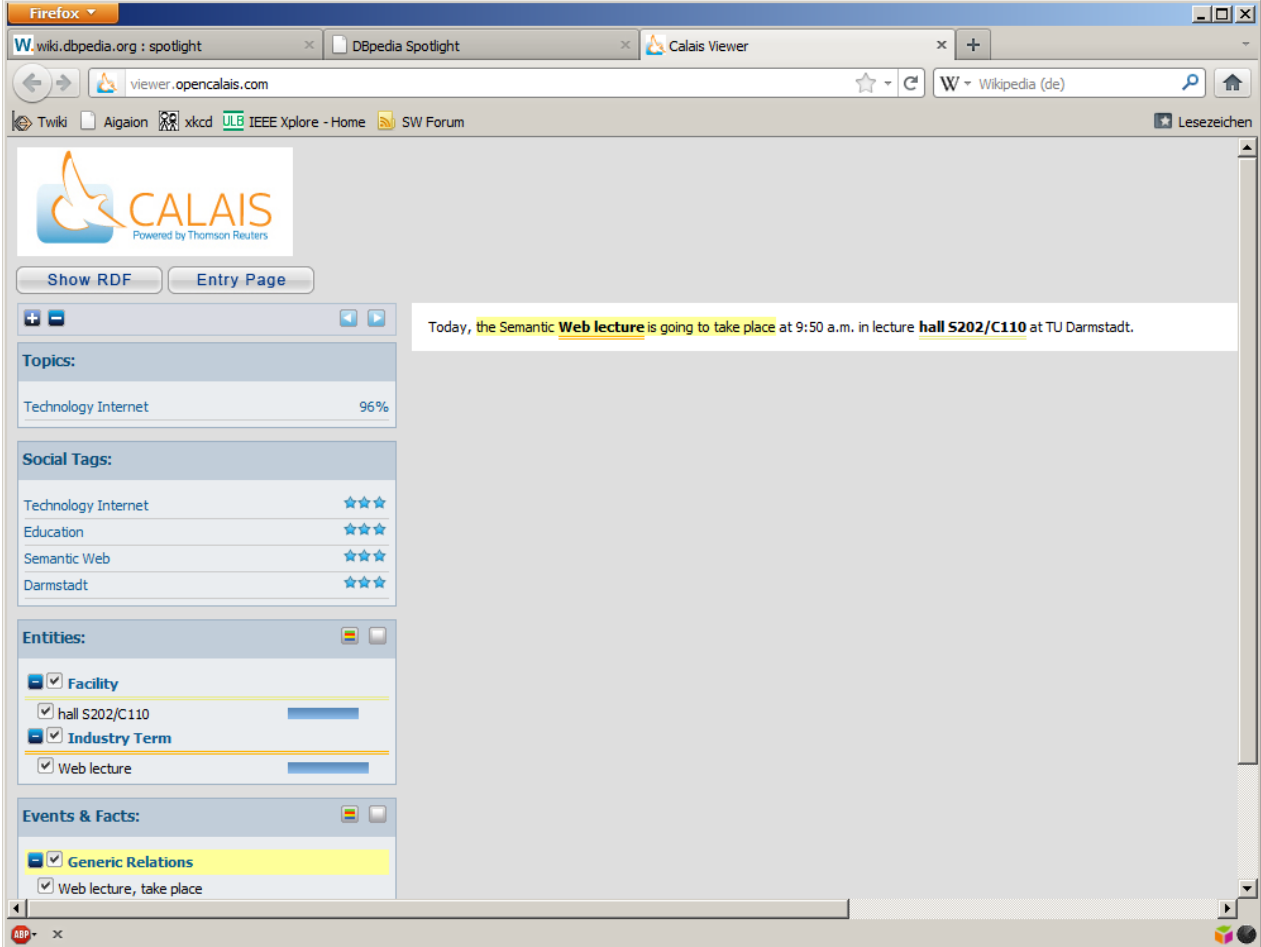
- This interface has been tested with Firefox 6.0.2 and Chromium 12.0.
- We have a cute [bookmarklet](#) that you should try out!
- DBpedia Spotlight is currently only available for **English**. However, it could be adapted for any language for which there is a DBpedia. [Contact us](#) if you'd like to help!

This demonstration uses the [DBpedia Spotlight iQuery Plugin v0.3](#) and the Web Service for [DBpedia Spotlight v0.5](#). For the latest versions, please visit: <http://spotlight.dbpedia.org>

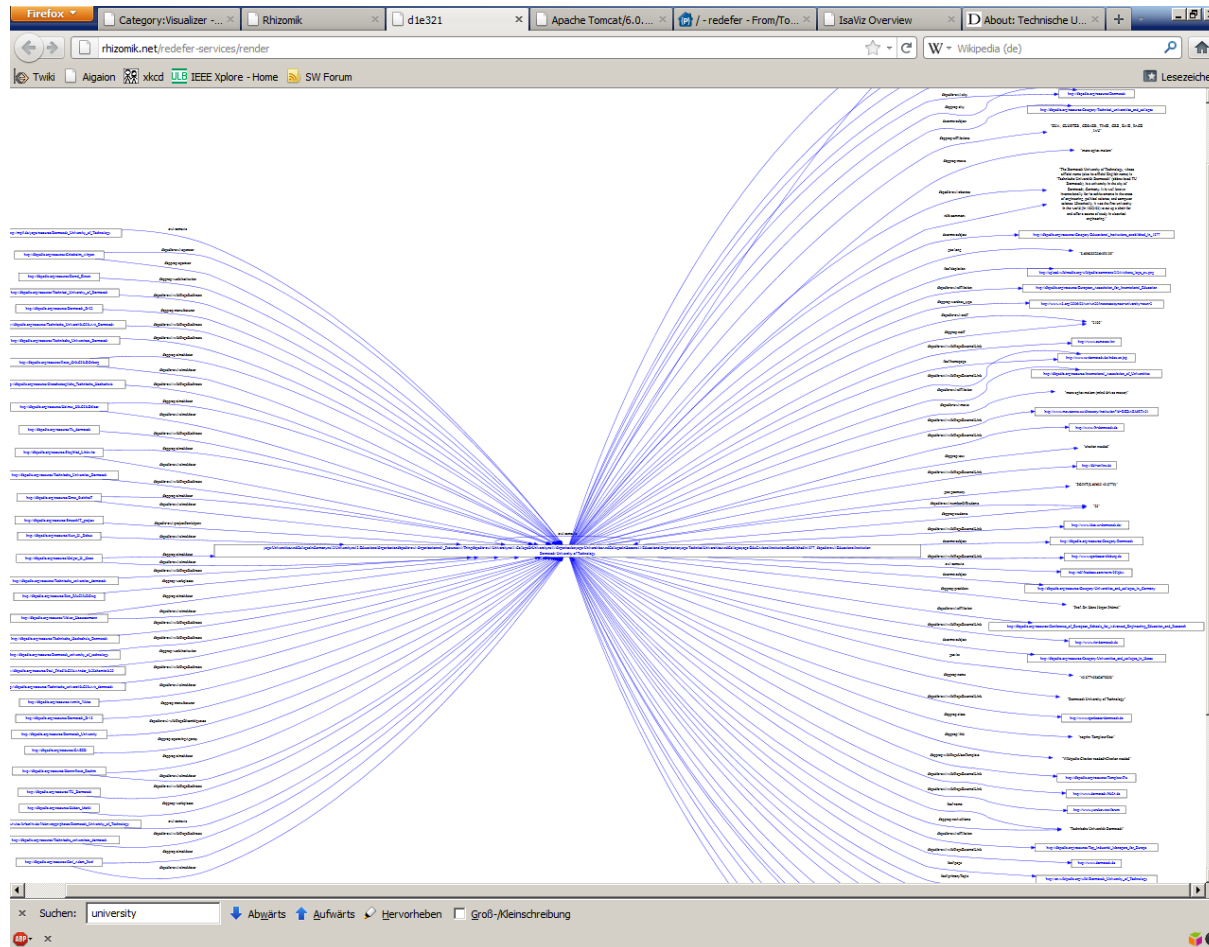
[http://dbpedia.org/resource/Darmstadt\\_University\\_of\\_Technology](http://dbpedia.org/resource/Darmstadt_University_of_Technology)

# Tagging

- Open Calais
  - Versucht auch Verbindungen zwischen Entitäten zu entdecken



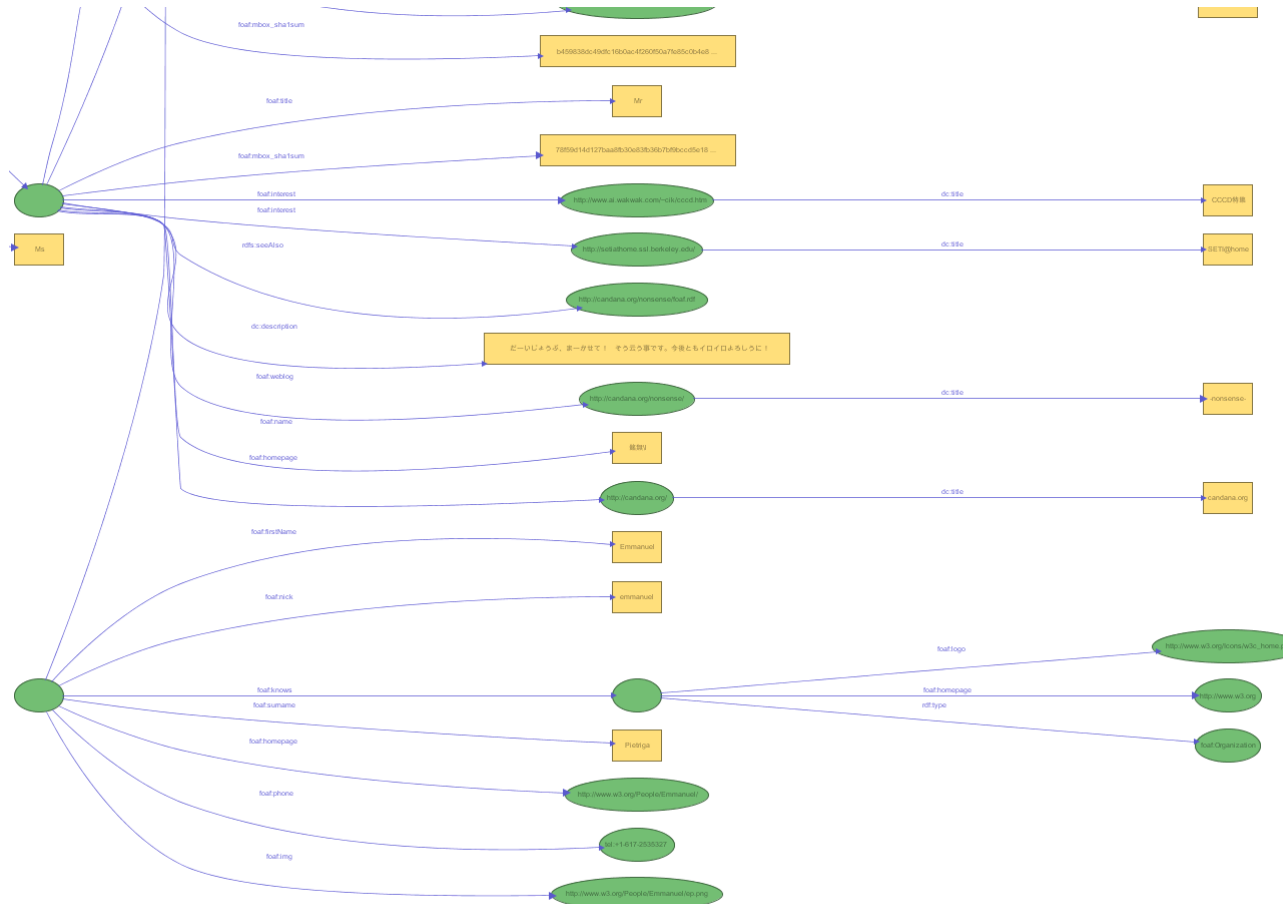
# Visualisierung



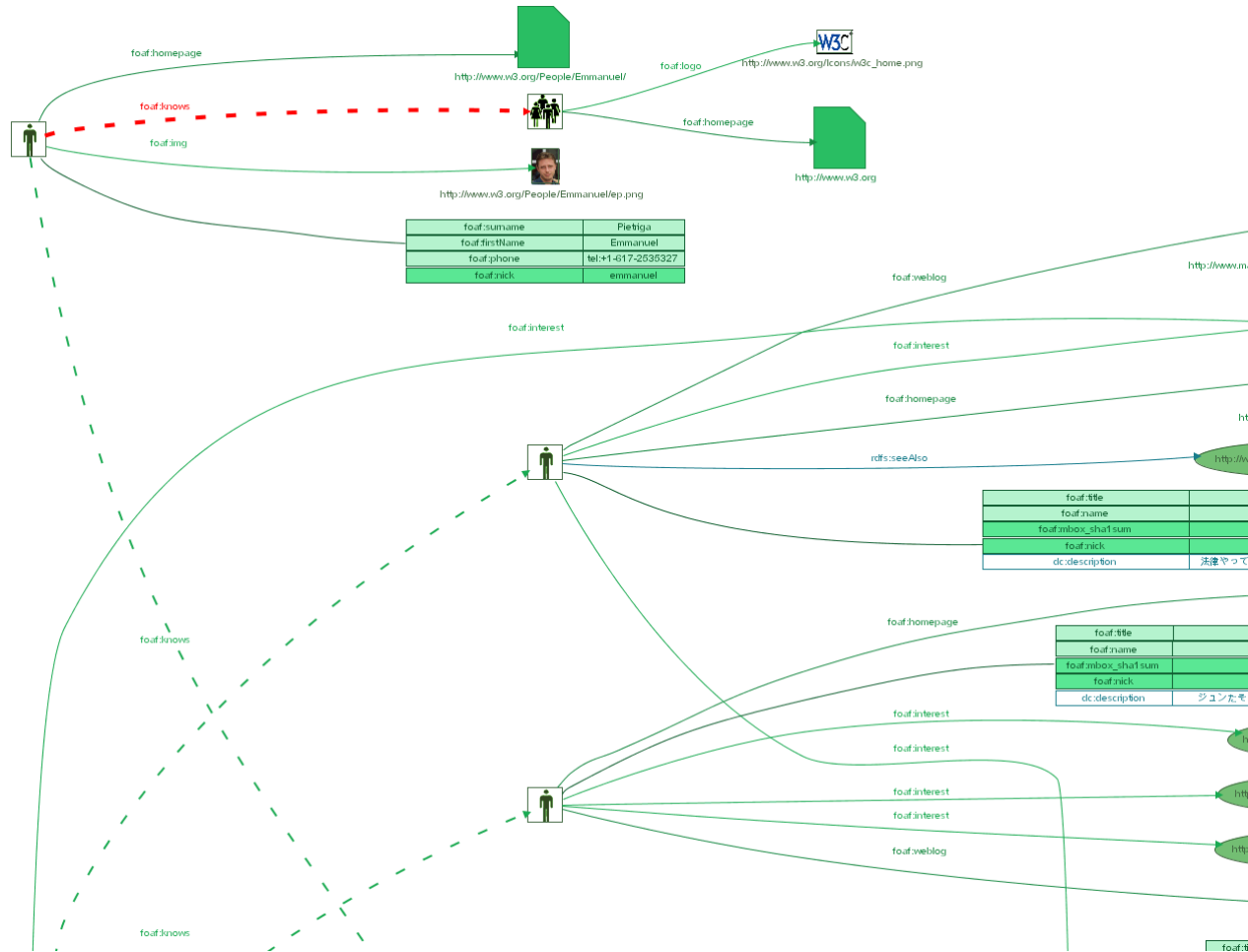


- Rhizomik ReDeFer
  - z.B. Darstellung von RDF als SVG
  - <http://rhizomik.net/html/redefer/>
- IsaViz
  - basiert auf GraphViz
  - diverse Visualisierungsformen
  - <http://www.w3.org/2001/11/IsaViz/>

# Visualisierung: IsaViz



# Visualisierung: IsaViz



# Zusammenfassung



- Programmierung von Semantic-Web-Anwendungen
  - mit direkten
  - und indirekten Modellen
- SPARQL-Unterstützung
  - auch für öffentliche Endpoints
- Reasoning ist in Programmier-Frameworks eingebaut
- Weitere nützliche Tools verfügbar
  
- Damit kann man intelligente Anwendungen bauen

# Vorlesung Semantic Web



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering