Vorlesung Semantic Web

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

Was bisher geschah

TECHNISCHE UNIVERSITÄT DARMSTADT

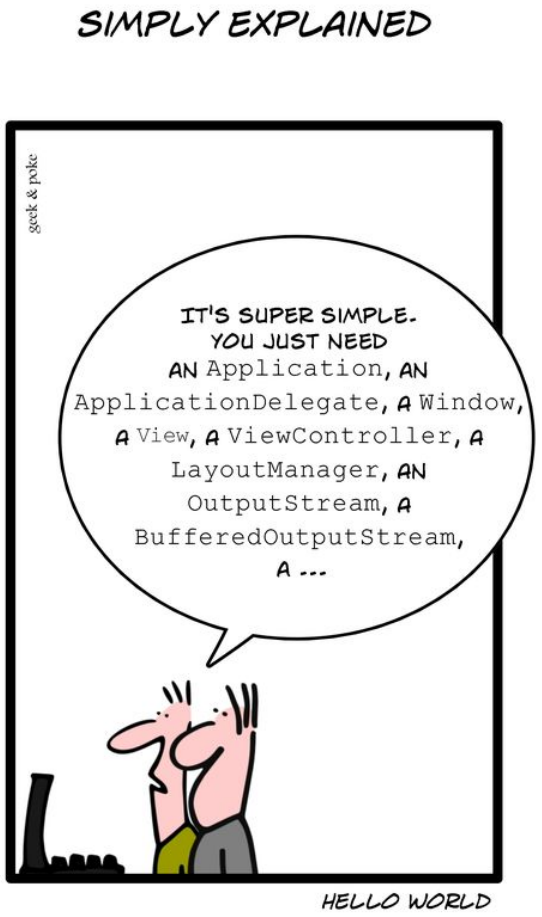
- Informationsrepräsentation mit RDF
- Schemata mit RDFS
- Bereitstellung der Information als Linked Data
- Ein wenig einfaches Reasoning
- Zugriff auf Information mit SPARQL

- Was uns noch fehlt
  - Wie kann man jetzt sinnvolle Anwendungen damit bauen?

Programmierung von Semantic Web Applikationen

TECHNISCHE UNIVERSITÄT DARMSTADT

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



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())
                    );

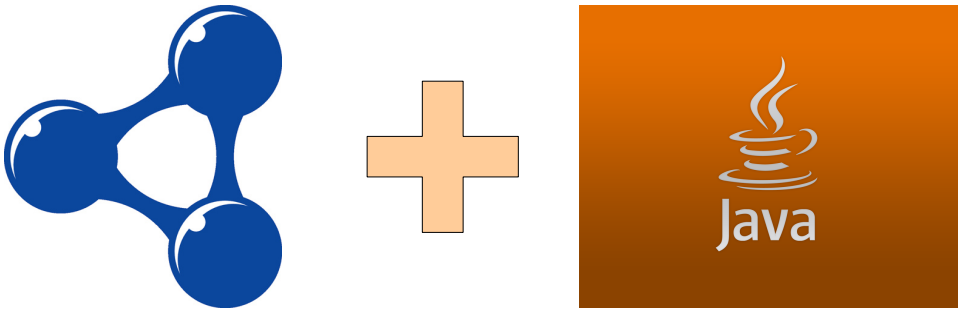
while(BR.ready()) {
  String triple = BR.readLine();
  StringTokenizer tokenizer = new StringTokenizer(triple, " ");
  String subject = tokenizer.nextToken();
  String predicate = tokenizer.nextToken();
  String object = tokenizer.nextToken();
  ...
}
```

Programmierung mit Frameworks

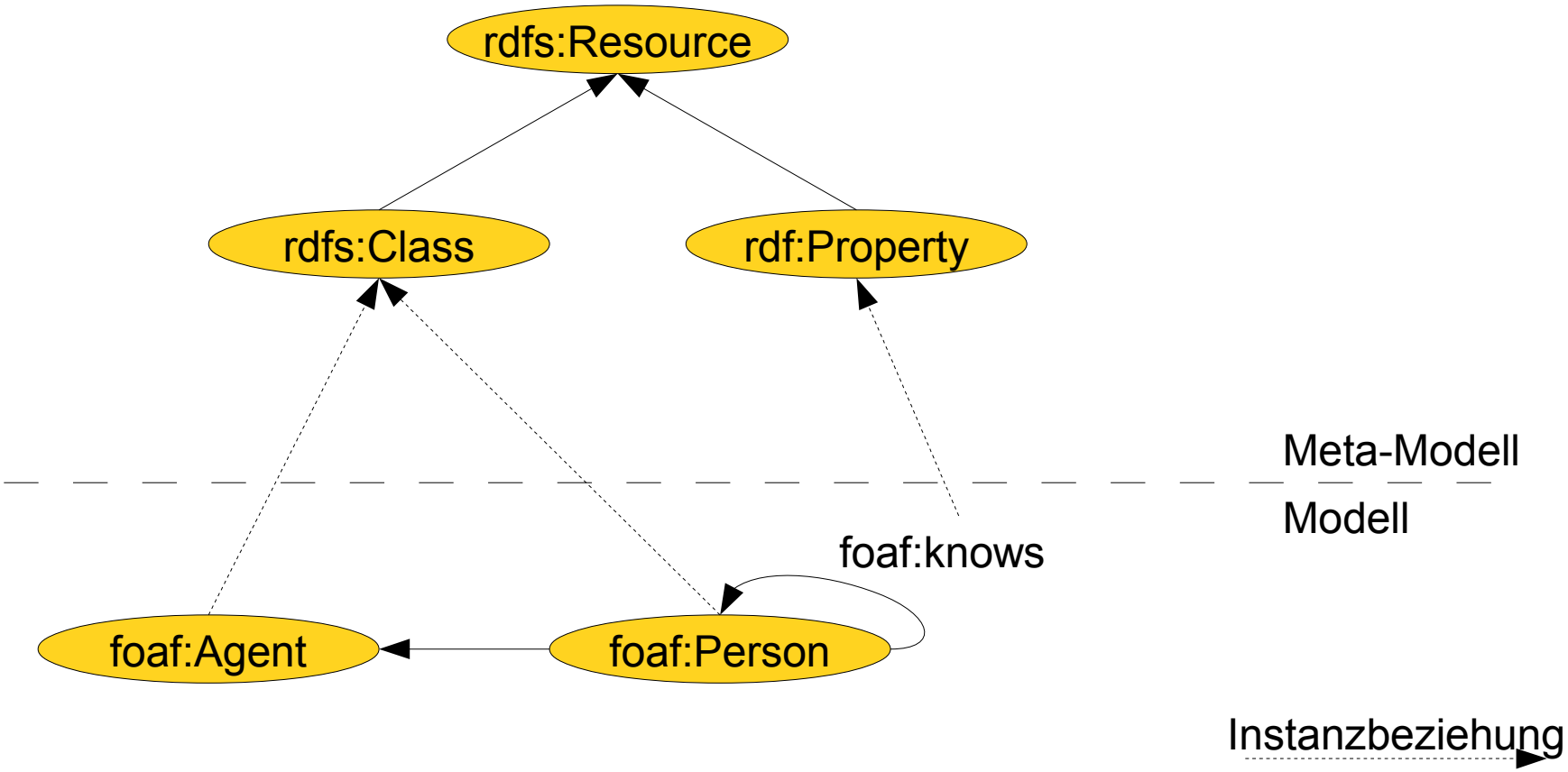
TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

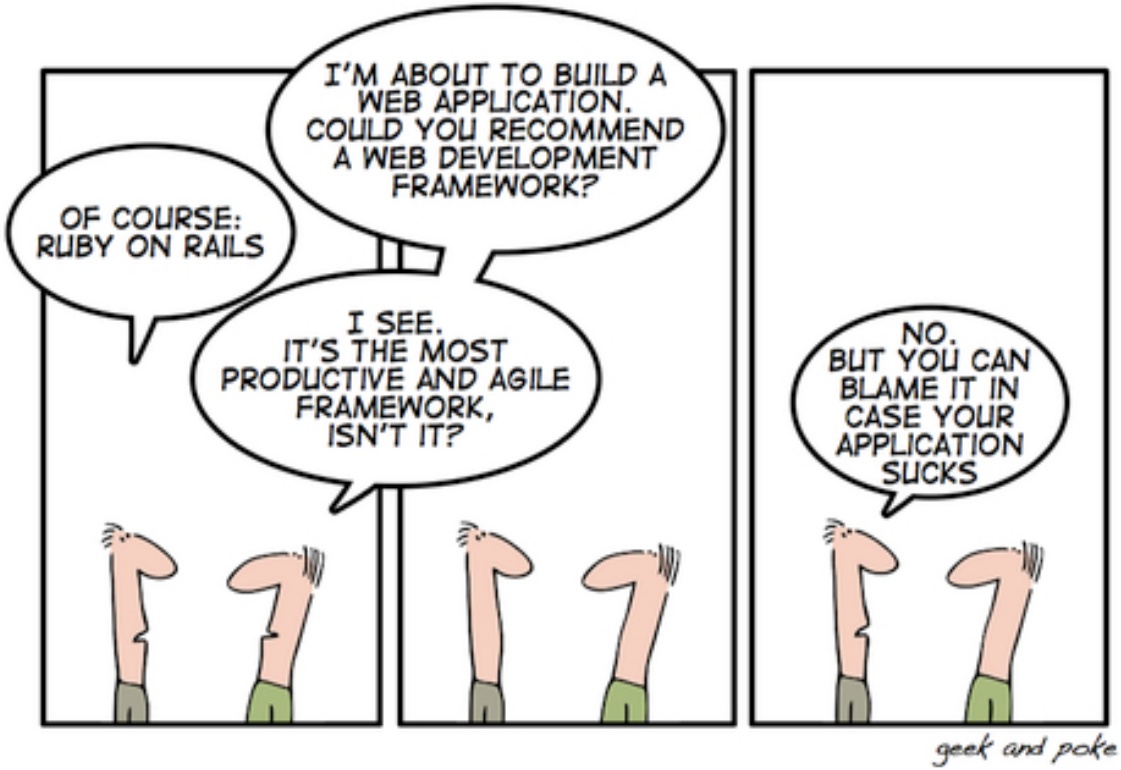
- Indirekte Programmierframeworks:
  - Java-Klassen entsprechen Klassen im RDF-Metamodell
  - z.B. für FOAF:

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

Direkte und indirekte Programmierframeworks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Man muss sich hier entscheiden



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

Direkte und indirekte Programmier-Frameworks

TECHNISCHE UNIVERSITÄT DARMSTADT

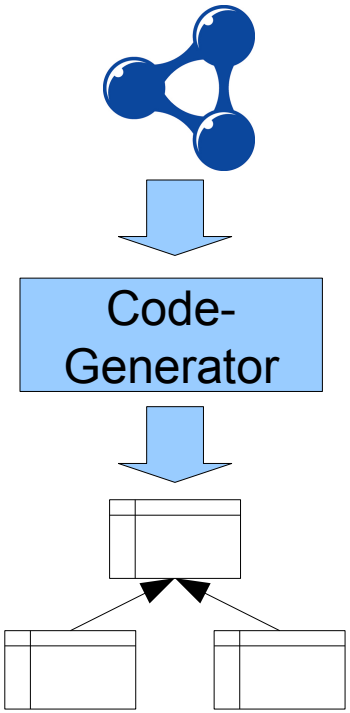
- Direkte Programmier-Frameworks
  - Gut lesbarer Code
  - Intuitiv
  - aber: nicht auf unbekannten Schemata anwendbar
  - und: das Schema bestimmt die Programmierung!

- Indirekte Programmier-Frameworks
  - weniger intuitiv
  - mehr Overhead
  - flexibler: Schema muss nicht bekannt sein
  - flexibler: das Schema kann sich weiterentwickeln

Direkte Programmier-Frameworks

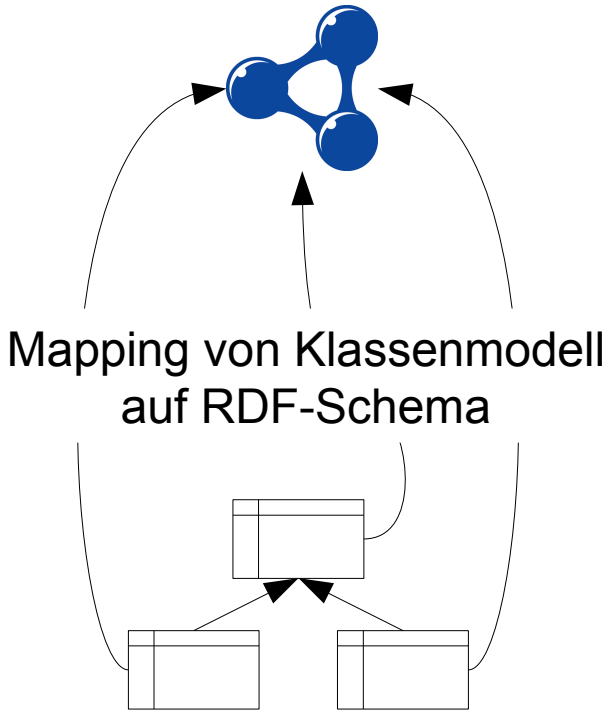
TECHNISCHE UNIVERSITÄT DARMSTADT

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



RDF-Schema

Klassenmodell



Beispiel: RDFReactor

TECHNISCHE UNIVERSITÄT DARMSTADT

- RDFReactor: Generiert Klassen aus Schema
- Diese können dann zur Verarbeitung von RDF genutzt werden

- Die Basis ist ein RDF2Go-Modell:

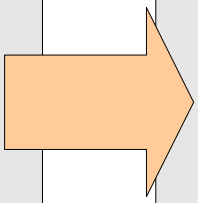
```
Model model = RDF2Go.getModelFactory().createModel();
```

Beispiel: RDFReactor und FOAF

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel: RDF von einer URI lesen und verarbeiten

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

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

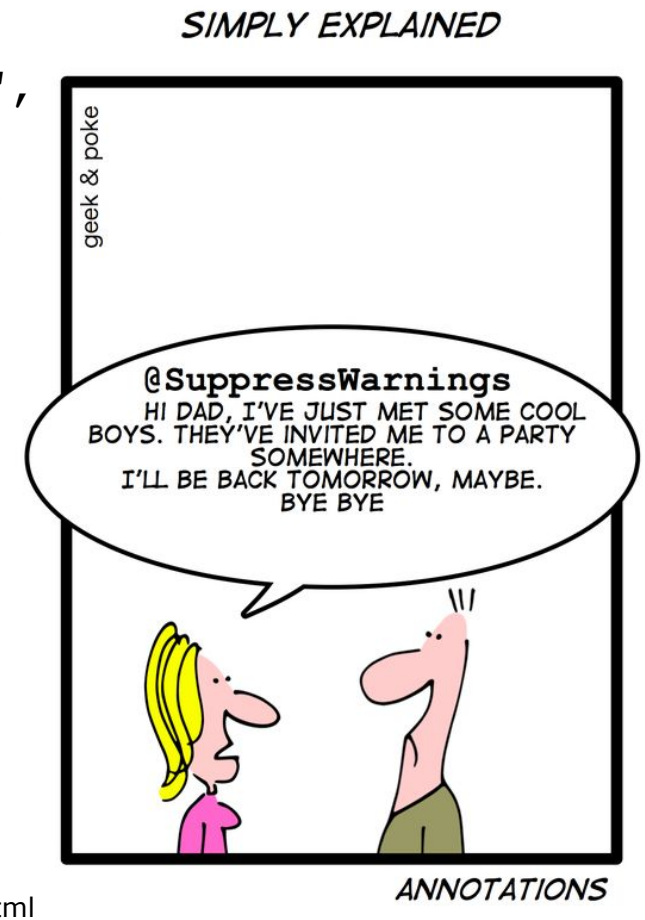
while(persons.hasNext() {
    Person[] friends = persons.next().getAllKnows();
    ...
}
```

Nutzung bestehender Klassen: Java Annotations

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel: otm-j

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



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

Nutzung bestehender Klassen: Java Annotations

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- Generierung von Code:
  - nur praktisch bei neuen Anwendungen
- Nutzung von Annotations
  - wenn Klassen im Source-Code vorliegen
  - und modifiziert werden können/dürfen

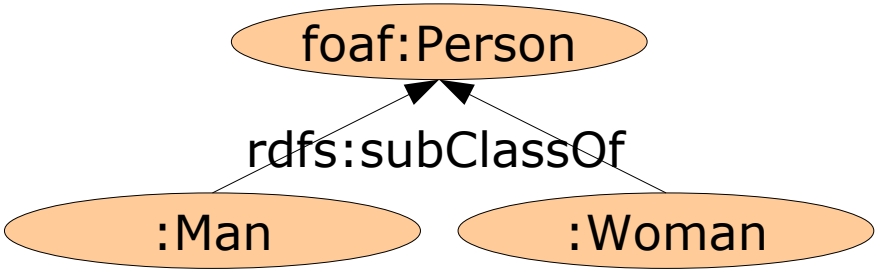
- Nicht-intrusive Varianten
  - Auslagern von Mappings in eigene Datei
  - z.B. ELMO: http://www.openrdf.org/doc/elmo/1.5/

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




```
public class Person {
  private enum Gender {male,female};
  private Gender gender;

  public Gender getGender() {
    return gender;
  }

  ...
}
```

Regelbasierte Abbildung RDF-Schema ↔ Klassenmodell

TECHNISCHE UNIVERSITÄT DARMSTADT

```
@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".
2nd 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
  - Datenbankanbindung für Persistenz
  - Unterstützung von SPARQL
  - Reasoning, Nutzung von Ontologien
  - Regel-Verarbeitung

Beispiel: Jena

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- Manipulation von Modellen

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

- Modelle überwachen

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

Beispiel: Jena + Reasoning

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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!

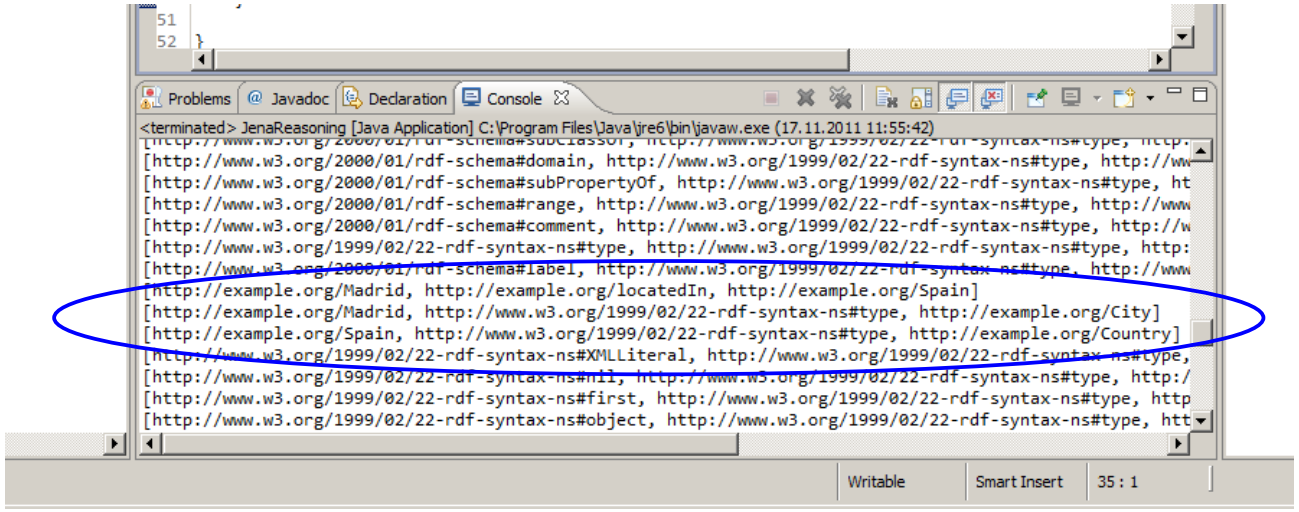
Beispiel: Jena + Reasoning

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das `reasoningModel` enthält jetzt alle abgeleiteten Fakten!

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

- Ausgabe:



Beispiel: Jena + Reasoning

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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
  - ...

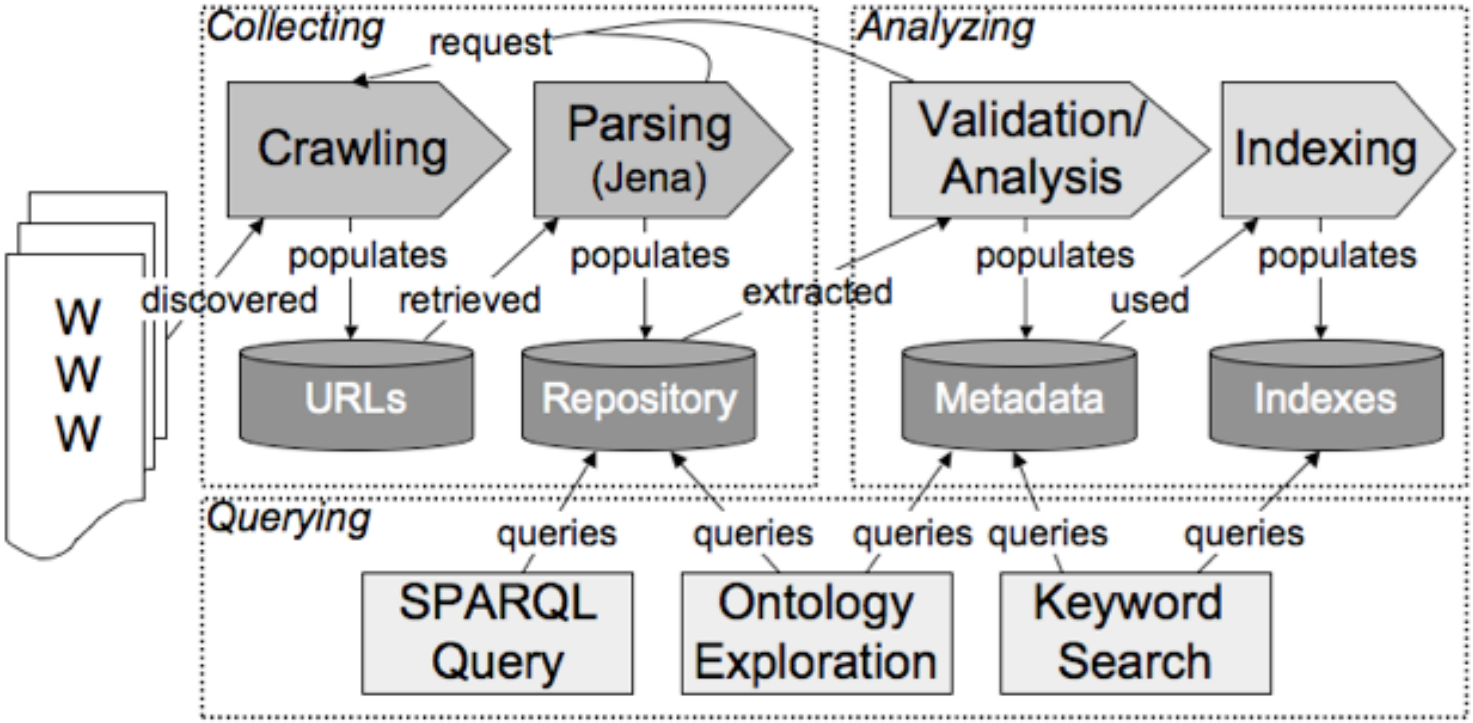
Suchen

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

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

Suchen

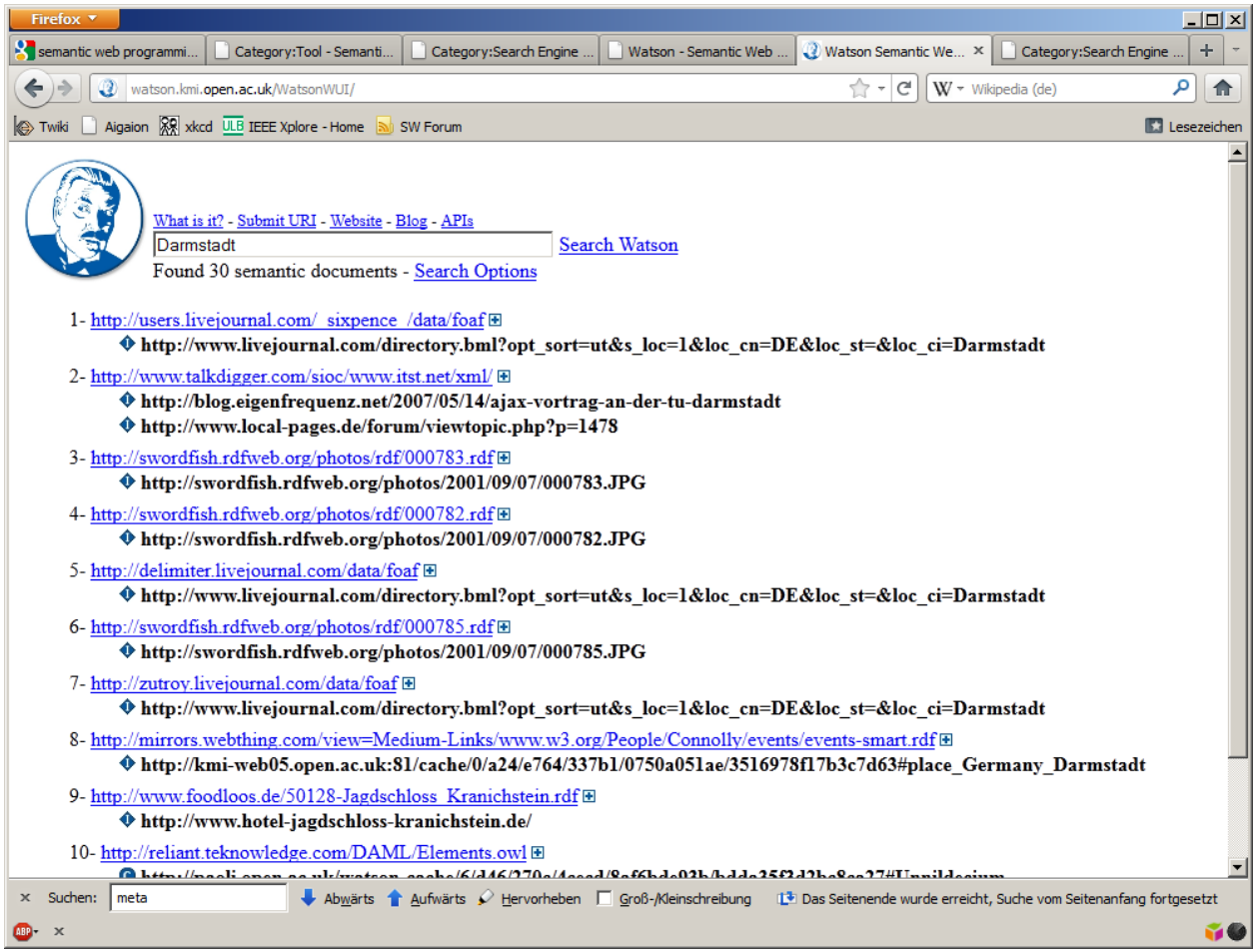
- Beispiel: RDF Watson – sucht RDF-Dokumente



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

Suchen

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



Suchen

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

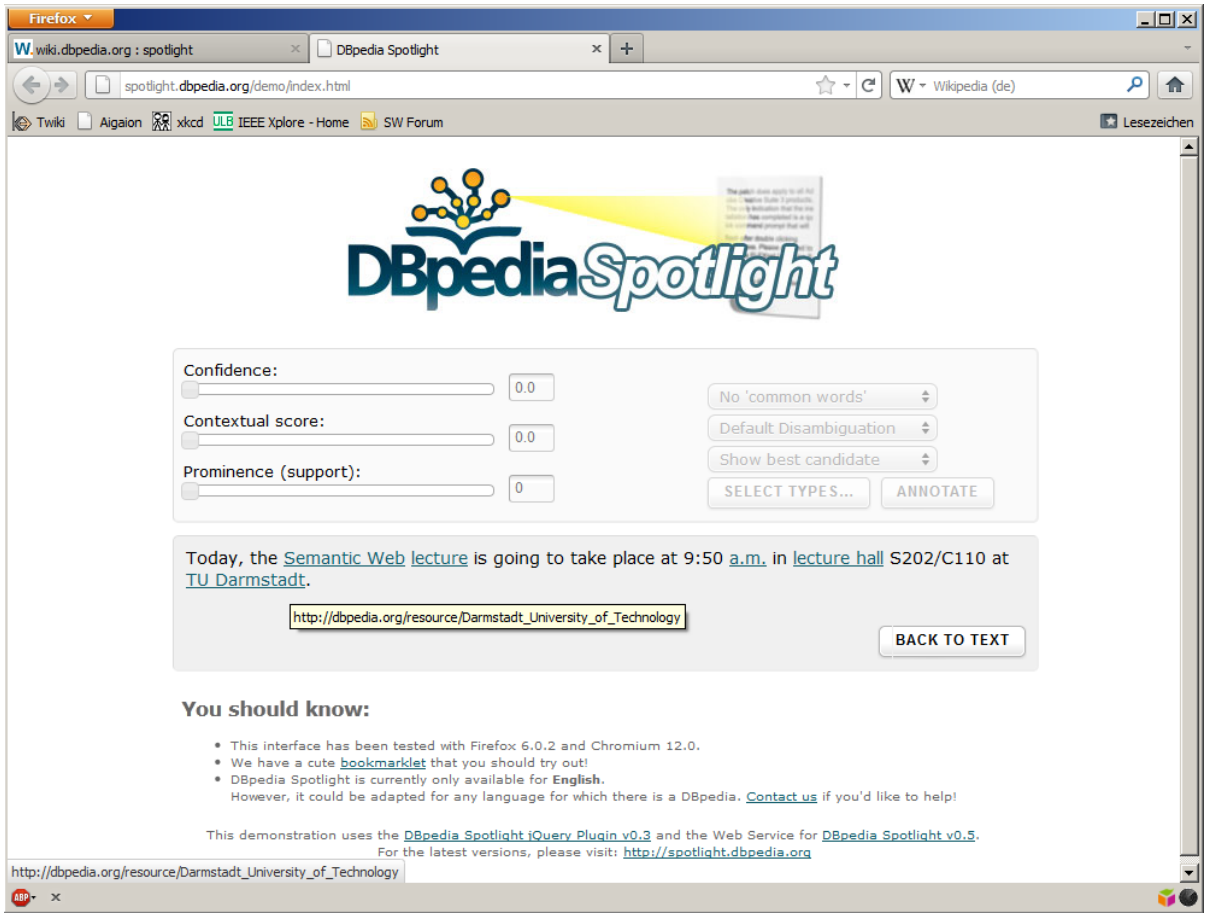
Suchen: Sindice

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

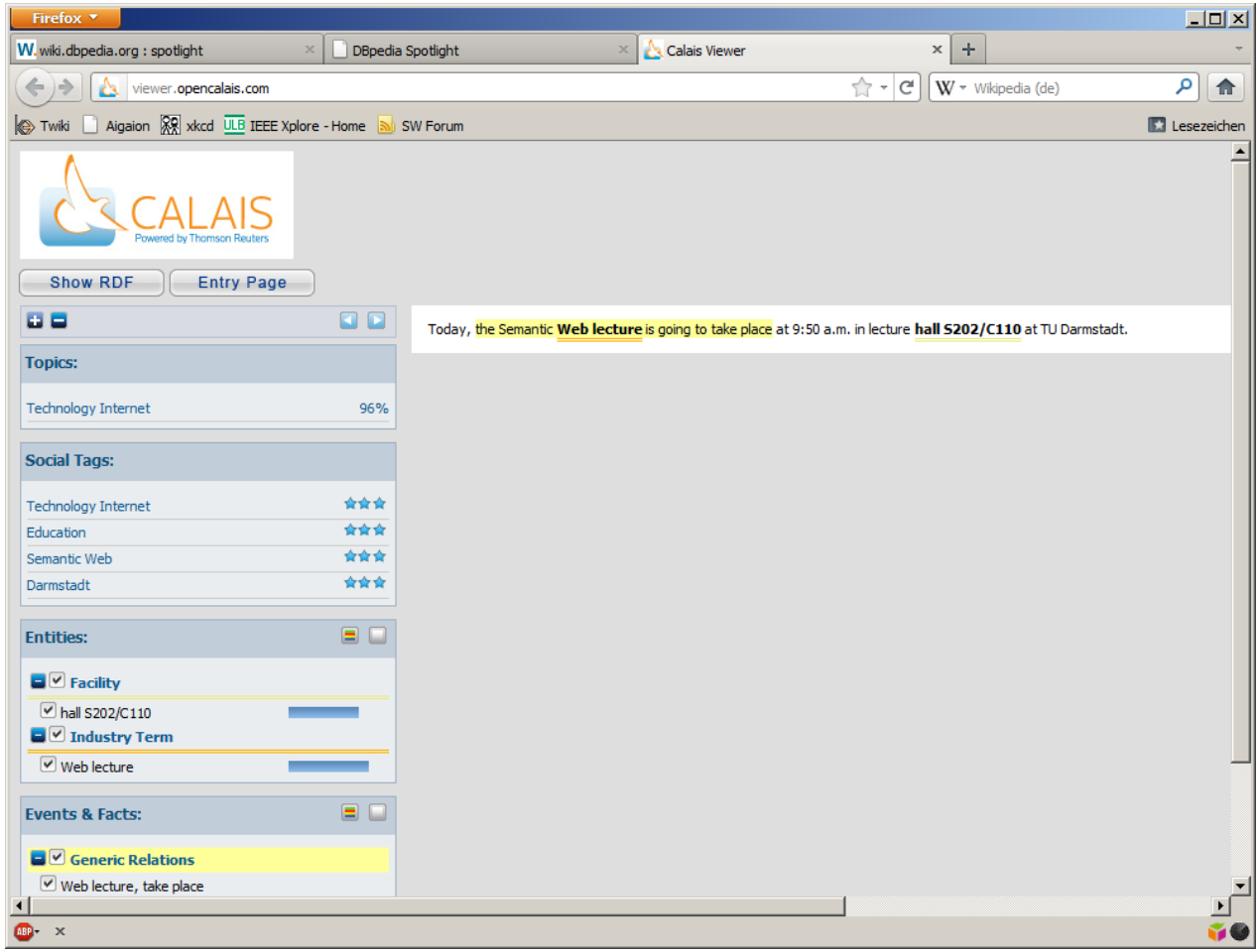
Tagging

- Text mit Links ins Semantic Web anreichern



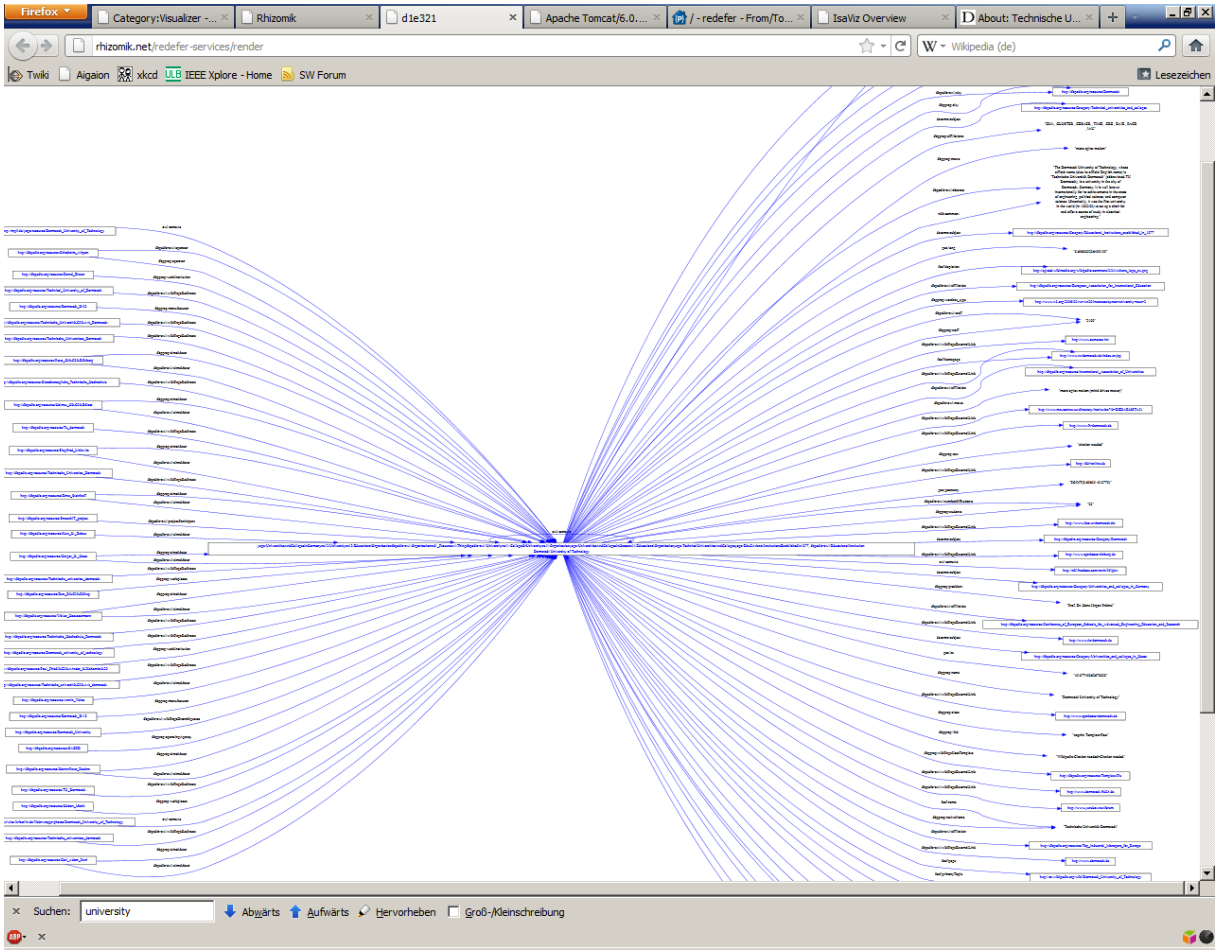
Tagging

- Open Calais
  - Versucht auch Verbindungen zwischen Entitäten zu entdecken



Visualisierung

TECHNISCHE UNIVERSITÄT DARMSTADT



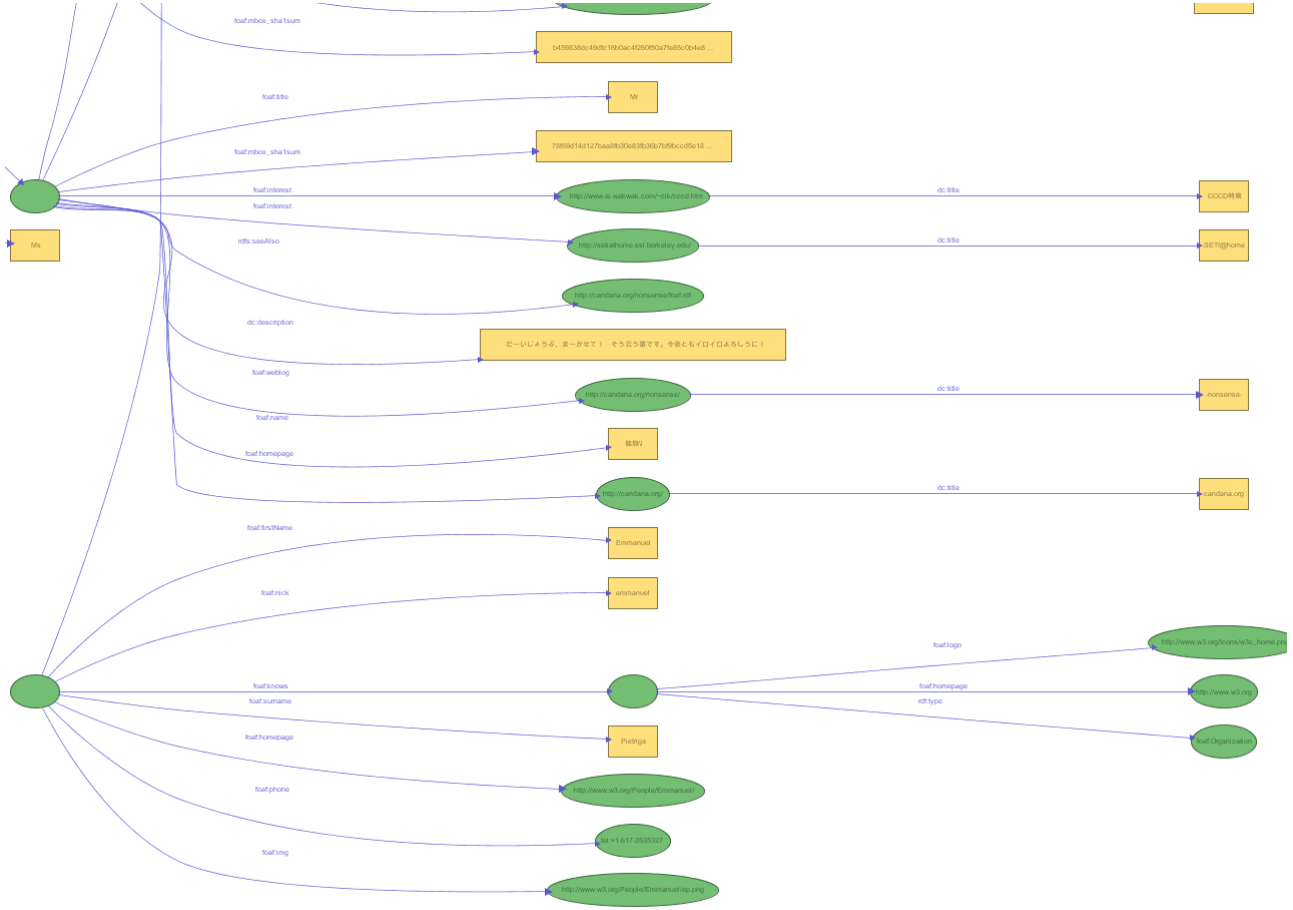
Visualisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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/

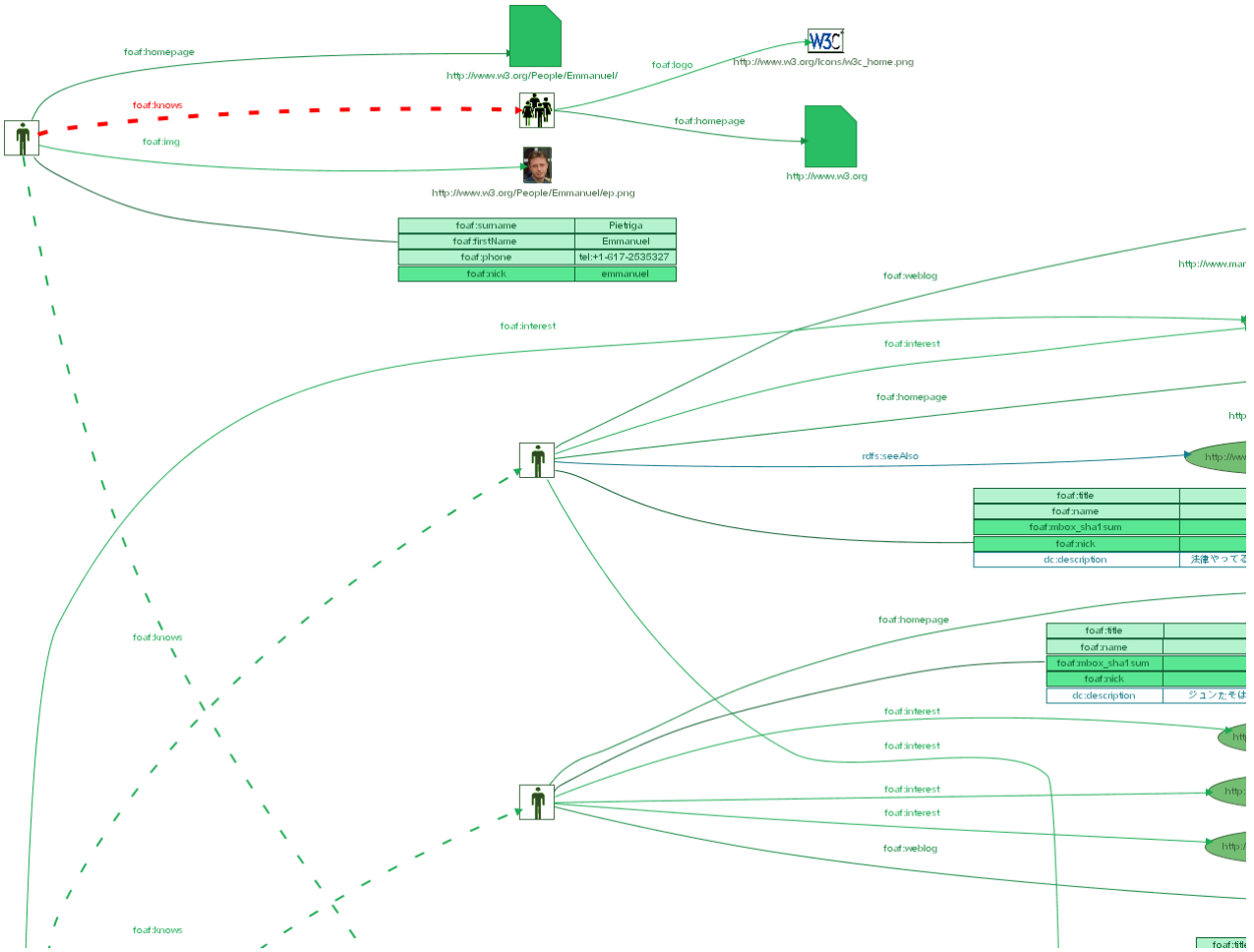
Visualisierung: IsaViz

TECHNISCHE UNIVERSITÄT DARMSTADT



Visualisierung: IsaViz

TECHNISCHE UNIVERSITÄT DARMSTADT



Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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