HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

Semantic Web

Übung 2

Markus Grandpré

18. Juni 2010

Betreuung
Steffen Schlönvoigt

Fakultät Informatik
Technische Informatik und Software Engineering

Inhaltsverzeichnis

1 Einführung in FOAF

Die folgenden Aufgaben dienen dazu, den RDF-Dialekt FOAF etwas näher kennenzulernen. Dazu soll ein einfaches FOAF-Profil erstellt und mit anderen Profilen verlinkt werden. Darüber hinaus sollen spezielle FOAF-Tools angewendet werden, um FOAF-Dateien zu generieren oder anzuzeigen. Zuletzt soll die FOAF-Spezifikation verwendet werden, um die erstellten FOAF-Profile zu erweitern.

1.1 Einfaches Profil

Das einfache Profil `Asterix_single` wurde zuerst mit dem FOAF-Generator [1] erstellt und dann manuell erweitert (z.B. Mailbox, Homepage, Image, Google-Map).

```
<!-- file: Asterix_single.rdf -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>

  <foaf:givenname>Asterix</foaf:givenname>
  <foaf:family_name>der Gallier</foaf:family_name>
  <foaf:mbox rdf:resource="mailto:asterix@gallisches.dorf"/>
  <foaf:homepage rdf:resource="http://www.gallisches.dorf"/>
  <foaf:depiction rdf:resource="http://www.comicportal.de/Asterix.jpg" />
  <foaf:based_near>
   <geo:Point geo:lat="48.619172" geo:long="-4.549026"/>
  </foaf:based_near>
 </foaf:Person>
</rdf:RDF>
```

Interessant ist dabei der Knoten `foaf:PersonalProfileDocument`. Dieser Knoten enthält ein leeres `foaf:about`-Attribut, dennoch wird dieses leere Attribut von einem Parser als Referenz auf sich selbst interpretiert:

```
<URL des Dokuments> rdf:type foaf:PersonalProfileDocument
```

die URL ist also vom Typ `foaf:PersonalProfileDocument`. Der Knoten `foaf:primaryTopic` sagt dann aus, um welche Resource bzw. Person es in diesem Dokument eigentlich geht, nämlich `#me`, wie es im folgenden `foaf:Person` dann mit dem Attribute `rdf:ID` noch einmal festgelegt wird.

Die RDF/FOAF-Datei `Asterix_single.rdf` wurde in das `semweb/`- Verzeichnis eines Apache2-Webservers kopiert, um den Inhalt der Datei mit dem Visualisierungsprogramm FOAFER [2] anzeigen lassen zu können:

`http://www.foafer.org/?file=http://emma.rz.uni-konstanz.de/semweb/Asterix_single.rdf`

[1] http://foaf.me
[2] http://www.foafer.org/

1.2 Verlinkung mit anderen Profilen

Zur Verlinkung mit anderen FOAF-Profilen wurde das oben dargestellte Profil `Asterix_-single` um `foaf:knows`-Tags erweitert:

```
...
<foaf:Person foaf:Person rdf:ID="me">
 ...
 <foaf:knows>
  <foaf:Person
   rdf:about="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf#me">
   <foaf:name>Obelix</foaf:name>
   <rdfs:seeAlso
    rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf"/>
  </foaf:Person>
 </foaf:knows>
 <foaf:knows>
  <foaf:Person
   rdf:about="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf#me">
   <foaf:name>Idefix</foaf:name>
   <rdfs:seeAlso
    rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf"/>
  </foaf:Person>
 </foaf:knows>
 ...
</foaf:Person>
...
```

Wenn man Beziehungen zu anderen Personen herstellen möchte, kann man das `foaf:knows`-Attribut verwenden, dessen `rdf:about`-Attribut den Identifier des Dokuments darstellt. Hierbei wird wieder von der Selbstreferenzierung innerhalb eines RDF-Dokumnets Gebrauch gemacht. Um einen wirklichen Verweis auf ein RDF-Dokument zu erzeugen, muß der RDFS-Tag `rdfs:seeAlso` benutzt werden, dessen `rdf:resource`-Attribut auf die jeweilige FOAF/RDF-Datei zeigt. Mit FOAFER erzeugt man somit einen anklickbaren Link, um das FOAF-Profil einer anderen Person aufzurufen. Um die Verlinkung zu demonstrieren, wurden verschiedene Profile erstellt, die mit dem Visualisierungs-Tool FOAFER dargestellt werden können:

- Profil “Asterix” (siehe Anhang A),
- Profil “Obelix” (siehe Anhang B),
- Profil “Idefix” (siehe Anhang C),
- Profil “Julius” (siehe Anhang D) und
- Profil “Nixalsverdrus” (siehe Anhang E).

1.3 Der FOAF-Generator “foaf-a-matic”

Das folgenden RDF/FOAF-Profil wurde mit dem FOAF-Generator “foaf-a-matic” [3] erstellt:

```
<!-- file: FOAF-a-MaticTest.rdf -->

<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:admin="http://webns.net/mvcb/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
  <admin:generatorAgent rdf:resource="http://www.ldodds.com/foaf/foaf-a-matic"/>
  <admin:errorReportsTo rdf:resource="mailto:leigh@ldodds.com"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Asterix der Gallier</foaf:name>
  <foaf:givenname>Asterix</foaf:givenname>
  <foaf:family_name>der Gallier</foaf:family_name>
  <foaf:mbox_sha1sum>48005874122b22271a9e557c9bac87b3d58c44e4</foaf:mbox_sha1sum>
  <foaf:homepage rdf:resource="www.gallisches.dorf"/>
  <foaf:depiction rdf:resource="http://www.comicportal.de/Asterix.jpg"/>
  <foaf:knows>
   <foaf:Person>
    <foaf:name>Obelix</foaf:name>
    <foaf:mbox_sha1sum>649d8e87a4cd182a0a059f1be06fc3bba3bdae77</foaf:mbox_sha1sum>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   ...
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Der Generator erzeugt FOAF-Dokumente unter Verwendung des erwähnten `foaf:PersonalProfileDocument`-Tags und erweitert diesen noch zusätzlich um das Attribut `foaf:maker`, in dem noch einmal festgehalten ist, wer dieses Dokument erstellt hat: `#me`. Zudem werden noch Attribute aus der MetaVocab-Spezifikation [4] verwendet:

- `admin:generatorAgent`: das Programm, welches das Profil erstellt hat
- `admin:admin:errorReportsTo`: die E-Mail-Adresse, an die man sich wenden kann, falls es während der Generierung des FOAF-Profils Fehler gegeben hat.

Interessant ist auch, wie der “foaf-a-matic” mit E-Mail-Adressen umgeht, indem er aus einer gegebenen E-Mail-Adresse einen Hashcode erzeugt. Damit wird es Spammern unmöglich gemacht, sich anhand er verschiedenen Beziehungen eines RDF-Dokuments zu anderen RDF-Dokumenten entlang zu parsen und verwertbare E-Mail-Adressen einzusammeln.

[3] http://www.ldodds.com/foaf/foaf-a-matic.de.html
[4] http://webns.net/mvcb/

1.4 Die FOAF-Spezifikation

Das `Obelix` Profil wurde mit Hilfe der FOAF-Spezifikation 0.97 [5] noch um den `foaf:interest`-Tag werweitert, mit welchem man einer FOAF-Person Interessen hinzufügen kann. Der verwendete `rdf:Description`- Tag mit dem Attribut `rdf:about` kann genutzt werden, die jeweiligen Interessen einer Person noch näher zu beschreiben. Der `dc:title` wird mit FOAFER dazu verwendet, dem Attribut `rdf:about` einen Titel zu geben, anstatt nur dessen URI darzustellen:

```
...
<foaf:interest>
 <rdf:Description rdf:about="http://www.biocrawler.com/w/images/c/ca/Asterix-Obelix.gif">
  <dc:title>Hinkelsteine hauen</dc:title>
 </rdf:Description>
</foaf:interest>
...
```

Beispiel:

http://www.foafer.org/?file=http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf

[5] http://xmlns.com/foaf/spec/

2 Implementierung eines einfachen FOAF Crawler

In Aufgabe 2 sollte ein einfacher FOAF Crawler mit Hilfe der in Java realisierten Jena Software Version 2.6.3 [6] implementiert werden. Dabei sollte

1. eine FOAF-Datei von der Festplatte gelesen und über alle im erzeugten Model enthaltenen Statements iteriert werden, um die Triple (S,P,O) auszugeben (vollständiger Quellcode in Anhang F).
2. eine FOAF-Datei von der Festplatte gelesen und über alle im erzeugten Model enthaltenen Statements iteriert werden, um die Triple (S,P,O) nur für gefundene Knoten vom Typ `foaf:Person` auszugeben (vollständiger Quellcode in Anhang G).
3. eine FOAF-Datei von einer URL gelesen werden und über alle im erzeugten Model enthaltenen Statements iteriert werden, um die Triple (S,P,O) nur für gefundene Knoten vom Typ `foaf:Person` auszugeben (vollständiger Quellcode in Anhang H).
4. eine FOAF-Datei von einer URL gelesen werden, um anhand der enthaltenen `foaf:knows`-Tags alle refernzierten FOAF/RDF-Dateien über ihre URI zu lesen und dort enthaltene Informationen auszugeben (vollständiger Quellcode in Anhang I).

2.1 FOAF-Dateien einlesen

Um eine FOAF-Datei von der Festplatte zu lesen, muss ein `Model`-Objekt erzeugt und die Klasse `FileManager` verwendet werden:

```
// create an empty model
Model model = ModelFactory.createDefaultModel();

// use the FileManager to find the input file
InputStream in = FileManager.get().open(inputFileName);

if (in == null)
{
   throw new IllegalArgumentException("File: " + inputFileName + " not found");
}

// read the FOAF/RDF file
model.read(in, null);
```

Nach dem das FOAF/RDF-Dokument gelesen wurde, ist es vollständig in dem Objekt der Klasse `Model` enthalten. Anstatt von der Festplatte kann eine FOAF-Datei auch von einer URI gelesen werden:

```
// get URL from popup window
String urlString = JOptionPane.showInputDialog("Enter URL:", JOptionPane.OK_OPTION);

// create an empty model
Model model = ModelFactory.createDefaultModel();

// data stream from url
InputStream in = null;

try
{
   // try to get data stream from url

   // compose url and get stream
   URL location = new URL(urlString);
```

[6] http://jena.sourceforge.net/

```
    in = location.openStream();

    if (in == null)
    {
 throw new IllegalArgumentException("URL: " + location + " not found");
    }
}
catch (Exception e)
{
    System.err.println("Failed to get RDF file from: " + urlString);
    System.exit(-1);
}

// read the RDF/XML file, add URL String
// object to apply base URL to model
model.read(in, location);
```

Ein Objekt der Klasse `JOptionPane` erzeugt dabei ein Popup-Fenster, welches zur Eingabemöglichkeit für die Zeichenkette der URL dient. Die Zeichenkette wird dann verwendet, um mit der Klasse `InputStream` einen Stream zu erzeugen und die FOAF-Datei zu lesen.

2.2 Inhalt von FOAF-Dateien anzeigen

Wenn das `Model`-Objekt erzeugt wurde, enthält es alle Informationen aus der eingelesenen FOAF-Datei. Um die enthaltenen Statements anzuzeigen, kann man ein `StmtIterator`-Objekt auf das Objekt der Klasse `Model` anwenden:

```
// list the statements in the Model
StmtIterator iter = model.listStatements();

// print out the predicate, subject and object of each statement
while (iter.hasNext())
{
    Statement stmt = iter.nextStatement(); // get next statement
    Resource subject = stmt.getSubject(); // get the subject
    Property predicate = stmt.getPredicate(); // get the predicate
    RDFNode object = stmt.getObject(); // get the object

    // filter annotations out
    if (!subject.isAnon())
    {
      System.out.print("S: " + subject.toString() + " ");
      System.out.print("P: " + predicate.toString() + " ");
      System.out.print("O: " + object.toString() + "\n");
    }
}
```

Die Klassen bzw. Interfaces in JENA stehen in folgender Beziehung zueinander:

- Das generische Interface `RDFNode` bildet einen beliebigen RDF-Knoten ab. Von diesem Interface erben die Interfaces `Resource` (Objekt) und `Literal` (Datenwert).
- Da an Stelle des Subjekts immer nur Resourcen stehen dürfen, kann man hierfür das Jena Interface `Resource` verwenden. An der Stelle des Objekts kann sowohl eine Ressource als auch ein Literal stehen, also verwendet man hier den generischeren RDFNode.
- Das Prädikat ist natürlich auch eine Ressource, allerdings eben eine spezielle, die eine Eigenschaft ausdrückt. Dafür gibt es das Jena-Interface `Property`, was eben wiederum vom Interface `Resource` abgeleitet ist.

Um einzelne Knoten nach dem Typ `foaf:Person` zu parsen, verwendet man zum einen den Namespace der gesuchen Resource (`http://xmlns.com/foaf/0.1/Person`) und

zum anderen das spezielle Iterator-Objekt der Klasse `ResIterator`, bei dessen Initialisierung man den Typ `foaf:Person` angeben kann:

```
// get all resources of type foaf:Person
Resource person = model.getResource("http://xmlns.com/foaf/0.1/Person");
ResIterator iter = model.listSubjectsWithProperty(RDF.type, person);

// for each person, show their foaf:name if known
Property name = model.getProperty("http://xmlns.com/foaf/0.1/name");

while (iter.hasNext())
{
   Resource subject = iter.nextResource(); // get subject
   Statement stmt = subject.getProperty(name);
   Property predicate = stmt.getPredicate(); // get predicate
   RDFNode object = stmt.getObject(); // get object

   // no need to filter annotation out this time
   System.out.print("S: " + subject.toString() + " ");
   System.out.print("P: " + predicate.toString() + " ");
   System.out.print("O: " + object.toString() + "\n");
}
```

Somit kann man mit dem Iterator über alle Objekt der Klasse `Resource` iterieren und erhält somit alle RDF-Staments (Triple).

2.3 Inhalt von referenzierten FOAF-Dateien anzeigen

Wie in 1.2 bereits beschrieben, werden `foaf:knows`-Knoten dazu verwendet, um Beziehungen zu anderen FOAF-Dateien herzustellen. Daher muss das eingelesene Model nach diesen Knoten durchsucht werden:

```
// init porperties to retrieve foaf:name and foaf:knows nodes
Property name = model.getProperty("http://xmlns.com/foaf/0.1/name");
Property knows = model.getProperty("http://xmlns.com/foaf/0.1/knows");

// init iterator to parse all nodes that
// have at leat one foaf:knows nodes as sub node
ResIterator r1 = model.listSubjectsWithProperty(knows);

// iterate foaf:knows nodes
while (r1.hasNext())
{
  // get foaf:name property of foaf:knows nodes
  Resource subject = r1.nextResource();
  Statement stmt = subject.getProperty(name);
  RDFNode object = stmt.getObject();
  ...
}
```

Sobald die Subjekte feststehen, welche `foaf:knows`-Knoten enthalten, können anhand dieser Objekte die einzelnen `foaf:knows`-Knoten geparst werden, um mit Hilfe der `Property`-Objekte die Objekte (der Name der referenzierten Person) und das `rdfs:seeAlso`-Attribut (die URI) zu erhalten:

```
// init property to retrieve rdfs:seeAlso nodes
Property seeAlso = model.getProperty("http://www.w3.org/2000/01/rdf-schema#seeAlso");

// init statement iterator to get URI
// of referenced RDF documents
StmtIterator s2 = subject1.listProperties(seeAlso);

// iterate all foaf:seeAlso nodes
while (s2.hasNext())
{
   // get object of foaf:seeAlso nodes
   Statement stmt2 = s2.nextStatement();
   RDFNode object2 = stmt2.getObject();

   // extract rdfs:seeAlso of found foaf:Person node
   // in order to get some data from corresponding RDF file
   String friendData = getDataFromRDF(object2.toString());

   // print result
   System.out.println(object.toString() + " knows "
       + object1.toString() + ", " + friendData);
}
```

In dieser zweiten Schleife werden dann die gefundenen Subjekte dazu verwendet, mit Hilfe des `rdfs:seeAlso`-Attributs die URL zu einem anderen FAOF-Dokument zu lesen. Diese URL wird dann verwendet, um aus den referenzierten Dokument einige Daten über die Person (z.B. die E-Mail-Adresse, die Homepage, ...) lesen zu können:

```
// get initial model from url string
Model model = getModelFromURL(urlString);

// set properties for foaf:mailbox, foaf:homepage and foaf:family_name nodes
Property mbox = model.getProperty("http://xmlns.com/foaf/0.1/mbox");
Property fName = model.getProperty("http://xmlns.com/foaf/0.1/family_name");
Property homepage = model.getProperty("http://xmlns.com/foaf/0.1/homepage");

// init iterator object for Resource objects
ResIterator r;

// get all family name nodes
r = model.listSubjectsWithProperty(fName);

// iterate family name nodes
while (r.hasNext())
{
  // get foaf:family_name data
  Resource subject = r.nextResource();
  Statement stmt = subject.getProperty(fName);
  RDFNode object = stmt.getObject();

  // compose data
  data = data + object.toString() + ", ";
}
```

Dazu werden wieder `Property`-Objekte verwendet, mit denen das jeweilige Model anhand eines Namespace durchsucht werden kann. Ergebnis bei Aufruf der URL des Asterix-Profils:

```
Asterix knows Obelix, der Gallier, mailto:obelix@gallisches.dorf, http://www.gallisches.dorf
Asterix knows Julius Caesar, Imperator, mailto:julius.caesar@forum.romanum,
 http://www.forum.romanum
Asterix knows Idefix, der Hund,
```

Anhang A FOAF-Profil `Asterix.rdf`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Asterix</foaf:name>
  <foaf:givenname>Asterix</foaf:givenname>
  <foaf:family_name>der Gallier</foaf:family_name>
  <foaf:mbox rdf:resource="mailto:asterix@gallisches.dorf"/>
  <foaf:homepage rdf:resource="http://www.gallisches.dorf"/>
  <foaf:depiction rdf:resource="http://www.comicportal.de/Asterix.jpg" />
  <foaf:based_near>
   <geo:Point geo:lat="48.619172" geo:long="-4.549026"/>
  </foaf:based_near>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf#me">
    <foaf:name>Obelix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf#me">
    <foaf:name>Idefix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Julius.rdf#me">
    <foaf:name>Julius Caesar</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Julius.rdf"/>
   </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Anhang B FOAF-Profil `Obelix.rdf`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Obelix</foaf:name>
  <foaf:givenname>Obelix</foaf:givenname>
  <foaf:family_name>der Gallier</foaf:family_name>
  <foaf:mbox rdf:resource="mailto:obelix@gallisches.dorf"/>
  <foaf:homepage rdf:resource="http://www.gallisches.dorf"/>
  <foaf:depiction rdf:resource="http://www.comicportal.de/Obelix.jpg" />
  <foaf:based_near>
   <geo:Point geo:lat="48.619172" geo:long="-4.549026"/>
  </foaf:based_near>
  <foaf:interest>
   <rdf:Description rdf:about="http://www.biocrawler.com/w/images/c/ca/Asterix-Obelix.gif">
    <dc:title>Hinkelsteine hauen</dc:title>
   </rdf:Description>
  </foaf:interest>
  <foaf:interest>
   <rdf:Description rdf:about="http://media.wz-interaktiv.de/galerie/52/6452/476x444_218129.jpg">
    <dc:title>Roemer hauen</dc:title>
   </rdf:Description>
  </foaf:interest>
  <foaf:interest>
   <rdf:Description rdf:about="http://ais.badische-zeitung.de/piece/01/4b/36/41/21706305.jpg">
    <dc:title>Wildschweine essen</dc:title>
   </rdf:Description>
  </foaf:interest>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Asterix.rdf#me">
    <foaf:name>Asterix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Asterix.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf">
    <foaf:name>Idefix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf"/>
   </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Anhang C FOAF-Profil `Idefix.rdf`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Idefix</foaf:name>
  <foaf:givenname>Idefix</foaf:givenname>
  <foaf:family_name>der Hund</foaf:family_name>
  <foaf:depiction rdf:resource="http://www.comicportal.de/Idefix.jpg" />
  <foaf:based_near>
   <geo:Point geo:lat="48.619172" geo:long="-4.549026"/>
  </foaf:based_near>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Asterix.rdf#me">
    <foaf:name>Asterix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Asterix.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf#me">
    <foaf:name>Obelix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Obelix.rdf"/>
   </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Anhang D FOAF-Profil `Julius.rdf`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Julius</foaf:name>
  <foaf:givenname>Gaius Iulius Caesar</foaf:givenname>
  <foaf:family_name>Imperator</foaf:family_name>
  <foaf:mbox rdf:resource="mailto:julius.caesar@forum.romanum"/>
  <foaf:homepage rdf:resource="http://www.forum.romanum"/>
  <foaf:depiction rdf:resource="http://www.asterix.com/encyclopedie/personnages/perso/r22.gif" />
  <foaf:based_near>
   <geo:Point geo:lat="41.903742" geo:long="12.487679"/>
  </foaf:based_near>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Nixalsverdrus.rdf#me">
    <foaf:name>Zenturio Nixalsverdrus</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Nixalsverdrus.rdf"/>
   </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Anhang E FOAF-Profil `Nixalsverdrus.rdf`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:name>Nixalsverdrus</foaf:name>
  <foaf:givenname>Marcus Nixalsverdus</foaf:givenname>
  <foaf:family_name>Zenturio</foaf:family_name>
  <foaf:mbox rdf:resource="mailto:marcus,nixalsverdrus@senatus.populusque.romanus"/>
  <foaf:homepage rdf:resource="http://www.senatus.populusque.romanus"/>
  <foaf:depiction rdf:resource="http://www.khalisi.com/comics/asterix/personae/nixalsverdrus.gif" />
  <foaf:based_near>
   <geo:Point geo:lat="41.903742" geo:long="12.487679"/>
  </foaf:based_near>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Julius.rdf#me">
    <foaf:name>Gaius Iulius Caesar</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Julius.rdf"/>
   </foaf:Person>
  </foaf:knows>
  <foaf:knows>
   <foaf:Person rdf:about="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf#me">
    <foaf:name>Idefix</foaf:name>
    <rdfs:seeAlso rdf:resource="http://emma.rz.uni-konstanz.de/semweb/Idefix.rdf"/>
   </foaf:Person>
  </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

Anhang F Quelltext `Aufgabe01.java`

```
package aufgabe01;

import java.io.InputStream;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import com.hp.hpl.jena.util.FileManager;

/**
 * Aufgabe 01
 *
 * Beginnen Sie damit, dass Sie das vorher heruntergeladene Profil in ein
 * Speicherbasiertes Modell einlesen und dann ueber alle Statements (der Jena
 * Ausdruck fÃ¼r Triple) iterieren, um dann deren Subjekt, Praedikat und Objekt
 * auszugeben.
 *
 */
public class Aufgabe01 {
  public static void main(String[] args) {
    // set input filename
    String inputFileName = "Asterix.rdf";

    // create an empty model
    Model model = ModelFactory.createDefaultModel();

    // use the FileManager to find the input file
    InputStream in = FileManager.get().open(inputFileName);

    if (in == null) {
      throw new IllegalArgumentException("File: " + inputFileName
          + " not found");
    }

    // read the RDF/XML file
    model.read(in, null);

    // list the statements in the Model
    StmtIterator iter = model.listStatements();

    // print out the predicate, subject and object of each statement
    while (iter.hasNext()) {
      Statement stmt = iter.nextStatement(); // get next statement
      Resource subject = stmt.getSubject(); // get the subject
      Property predicate = stmt.getPredicate(); // get the predicate
      RDFNode object = stmt.getObject(); // get the object

      // filter annotations out
      if (!subject.isAnon()) {
        System.out.print("S: " + subject.toString() + " ");
        System.out.print("P: " + predicate.toString() + " ");
        System.out.print("O: " + object.toString() + "\n");
      }
    }
  }
}
```

Anhang G Quelltext `Aufgabe02.java`

```java
package aufgabe02;

import java.io.InputStream;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.ResIterator;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.util.FileManager;
import com.hp.hpl.jena.vocabulary.RDF;

/**
 * Aufgabe 02
 *
 * Erweitern Sie Ihr Programm derart, dass nur noch Triple ausgegeben werden,
 * die eine Person als Subjekt haben.
 *
 */
public class Aufgabe02 {
  public static void main(String[] args) {
    // set input filename
    String inputFileName = "Asterix.rdf";
    // String inputFileName = "TimBernersLee.rdf";

    // create an empty model
    Model model = ModelFactory.createDefaultModel();

    // use the FileManager to find the input file
    InputStream in = FileManager.get().open(inputFileName);

    if (in == null) {
      throw new IllegalArgumentException("File: " + inputFileName
          + " not found");
    }

    // read the RDF/XML file
    model.read(in, null);

    // get all resources of type foaf:Person
    Resource person = model.getResource("http://xmlns.com/foaf/0.1/Person");
    ResIterator iter = model.listSubjectsWithProperty(RDF.type, person);

    // for each person, show their foaf:name if known
    Property name = model.getProperty("http://xmlns.com/foaf/0.1/name");

    while (iter.hasNext()) {
      Resource subject = iter.nextResource(); // get subject
      Statement stmt = subject.getProperty(name);
      Property predicate = stmt.getPredicate(); // get predicate
      RDFNode object = stmt.getObject(); // get object

      // no need to filter annotation out this time
      System.out.print("S: " + subject.toString() + " ");
      System.out.print("P: " + predicate.toString() + " ");
      System.out.print("O: " + object.toString() + "\n");
    }
  }
}
```

Anhang H Quelltext `Aufgabe03.java`

```
package aufgabe03;

import java.io.InputStream;
import java.net.URL;

import javax.swing.JOptionPane;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.ResIterator;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.vocabulary.RDF;

/**
 * Aufgabe 03
 *
 * Versuchen Sie auch, die Informationen von ihrer (oder einer beliebigen
 * anderen FOAF URL) anstatt direkt aus der lokalen Datei zu lesen.
 */
public class Aufgabe03 {
  public static void main(String[] args) {
    // get URL from popup window
    String urlString = JOptionPane.showInputDialog("Enter URL:",
        JOptionPane.OK_OPTION);

    // create an empty model
    Model model = ModelFactory.createDefaultModel();

    // data stream from url
    InputStream in = null;

    try {
      // try to get data stream from url

      // compose url and get stream
      URL location = new URL(urlString);
      in = location.openStream();

      if (in == null) {
        throw new IllegalArgumentException("URL: " + location
            + " not found");
      }
    } catch (Exception e) {
      System.err.println("Failed to get RDF file from: " + urlString);
      System.exit(-1);
    }

    // read the RDF/XML file, add URL String
     // object to apply base URL to model
    model.read(in, urlString);

    // get all resources of type foaf:Person
    Resource person = model.getResource("http://xmlns.com/foaf/0.1/Person");
    ResIterator iter = model.listSubjectsWithProperty(RDF.type, person);

    // for each person, show their foaf:name if known
    Property name = model.getProperty("http://xmlns.com/foaf/0.1/name");

    while (iter.hasNext()) {
      Resource subject = iter.nextResource(); // get subject
      Statement stmt = subject.getProperty(name);
      Property predicate = stmt.getPredicate(); // get predicate
      RDFNode object = stmt.getObject(); // get object

      // no need to filter annotation out this time
      System.out.print("S: " + subject.toString() + " ");
```

```
      System.out.print("P: " + predicate.toString() + " ");
      System.out.print("O: " + object.toString() + "\n");
    }
  }
}
```

Anhang I Quelltext `Aufgabe04.java`

```
package aufgabe04;

import java.io.InputStream;
import java.net.URL;

import javax.swing.JOptionPane;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.ResIterator;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.StmtIterator;

/**
 * Aufgabe 04
 *
 * Erweitern Sie das Programm, so dass es versucht Informationen zu Ihren
 * Freunden (oder den Freunden eines anderen Profils) abzurufen. Geben Sie dann
 * etwa Vor- und Nachnamen dieser Freunde aus.
 */
public class Aufgabe04 {
  /**
   * This method returns a Jena Model representation of an RDF file that has
   * been retrieved from a URL.
   *
   * @param urlString
   *            to RDF file
   * @return Jena Model representation of an RDF file
   */
  public static Model getModelFromURL(String urlString) {
    // create default model
    Model model = ModelFactory.createDefaultModel();

    // data stream from url
    InputStream in = null;

    try {
      // try to get data stream from url

      // compose url and get stream
      URL location = new URL(urlString);
      in = location.openStream();

      if (in == null) {
        throw new IllegalArgumentException("URL: " + location
            + " not found");
      }
    } catch (Exception e) {
      // failed to get stream
      System.err.println("Failed to get RDF file from: " + urlString);
      System.exit(-1);
    }

    // read the RDF/XML file
    model.read(in, urlString);

    return model;
  }

  /**
   * This method is used to return some data (e.g. e-mail adress, homepage,
   * ...) from RDF-file at a specified URL.
   *
   * @param urlString
   *            to RDF file
   * @return data from RDF file
```

```
 */
public static String getDataFromRDF(String urlString) {
  // initi data String object
  String data = "";

  // get initial model from url string
  Model model = getModelFromURL(urlString);

  // set properties for foaf:mailbox, foaf:homepage and foaf:family_name
  // nodes
  Property mbox = model.getProperty("http://xmlns.com/foaf/0.1/mbox");
  Property fName = model
      .getProperty("http://xmlns.com/foaf/0.1/family_name");
  Property homepage = model
      .getProperty("http://xmlns.com/foaf/0.1/homepage");

  // init iterator object for Resource objects
  ResIterator r;

  // get all family name nodes
  r = model.listSubjectsWithProperty(fName);

  // iterate family name nodes
  while (r.hasNext()) {
    // get foaf:family_name data
    Resource subject = r.nextResource();
    Statement stmt = subject.getProperty(fName);
    RDFNode object = stmt.getObject();

    // compose data
    data = data + object.toString() + ", ";
  }

  // get all foaf:mbox nodes
  r = model.listSubjectsWithProperty(mbox);

  // iterate all foaf:mailbox nodes
  while (r.hasNext()) {
    // get foaf:mbox data
    Resource subject = r.nextResource();
    Statement stmt = subject.getProperty(mbox);
    RDFNode object = stmt.getObject();

    // compose data
    data = data + object.toString() + ", ";
  }

  // get all foaf:homepage nodes
  r = model.listSubjectsWithProperty(homepage);

  // iterate all foaf:hompage nodes
  while (r.hasNext()) {
    // get all foaf:homepage data
    Resource subject = r.nextResource();
    Statement stmt = subject.getProperty(homepage);
    RDFNode object = stmt.getObject();

    // compose data
    data = data + object.toString();
  }

  return data;
}

public static void main(String[] args) {
  // get URL from popup window
  String urlString = JOptionPane.showInputDialog("Enter URL:",
      JOptionPane.OK_OPTION);

  // get initial model from url string
  Model model = getModelFromURL(urlString);
```

```
    // init porperties to retrieve foaf:name and foaf:knows nodes
    Property name = model.getProperty("http://xmlns.com/foaf/0.1/name");
    Property knows = model.getProperty("http://xmlns.com/foaf/0.1/knows");

    // init iterator to parse all nodes that
    // have at leat one foaf:knows nodes as sub node
    ResIterator r1 = model.listSubjectsWithProperty(knows);

    // iterate foaf:knows nodes
    while (r1.hasNext()) {
      // get foaf:name property of foaf:knows nodes
      Resource subject = r1.nextResource();
      Statement stmt = subject.getProperty(name);
      RDFNode object = stmt.getObject();

      // init iterator
      StmtIterator s = subject.listProperties(knows);

      // iterate all foaf:knows nodes
      while (s.hasNext()) {
        // get foaf:name property of foaf:knows nodes
        Statement stmt1 = s.nextStatement().getProperty(name);
        Resource subject1 = stmt1.getSubject();
        RDFNode object1 = stmt1.getObject();

        Property seeAlso = model.getProperty("http://www.w3.org/2000/01/rdf-schema#seeAlso");

        StmtIterator s2 = subject1.listProperties(seeAlso);

        // iterate all foaf:seeAlso nodes
        while (s2.hasNext()) {
          // get object of foaf:seeAlso nodes
          Statement stmt2 = s2.nextStatement();
          RDFNode object2 = stmt2.getObject();

          // extract rdfs:seeAlso of found foaf:Person node
          // in order to get some data from corresponding RDF file
          String friendData = getDataFromRDF(object2.toString());

          // print result
          System.out.println(object.toString() + " knows "
              + object1.toString() + ", " + friendData);

        }
      }
    }
  }
}
```