Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

Semantic Web Advanced Topics

Ontology Languages

Knut Hinkelmann

Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

Outline of the Lecture

- Topic Maps
- F-Logic
- RDF/OWL

Ontologie-Sprachen im WWW

Topic Maps
(ISO Standard)

T-A-O

Topics

Associations

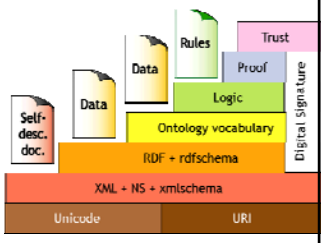
Occurences

F-Logic

- F-Logic: Objekt-orienterte logische Programmierung
- WSML erweitert F-Logic um Elemente von Description Logic

RDF/OWL
(W3C Standard)

Basiert auf Description Logic



Topic Maps

- Topic Maps …
  - are a formalism for the representation and exchange of knowledge
  - can represent meta knowledge about information resources (documents, images, web pages …)
  - represent knowledge in the form of semantic networks
- Objective of topic maps
  - enhance searching and browsing for information
  - increase quality and speed of finding information
  - integration various information sources

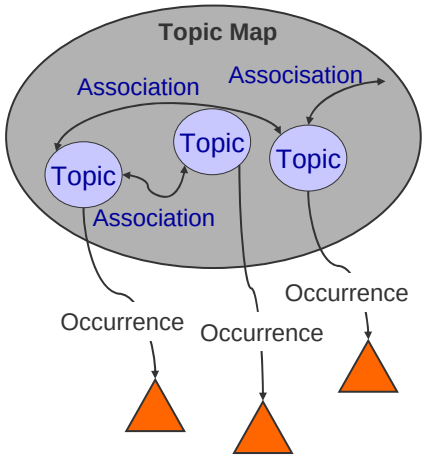
Basic Constructs of Topic Maps

Topic maps represent domain knowledge using…

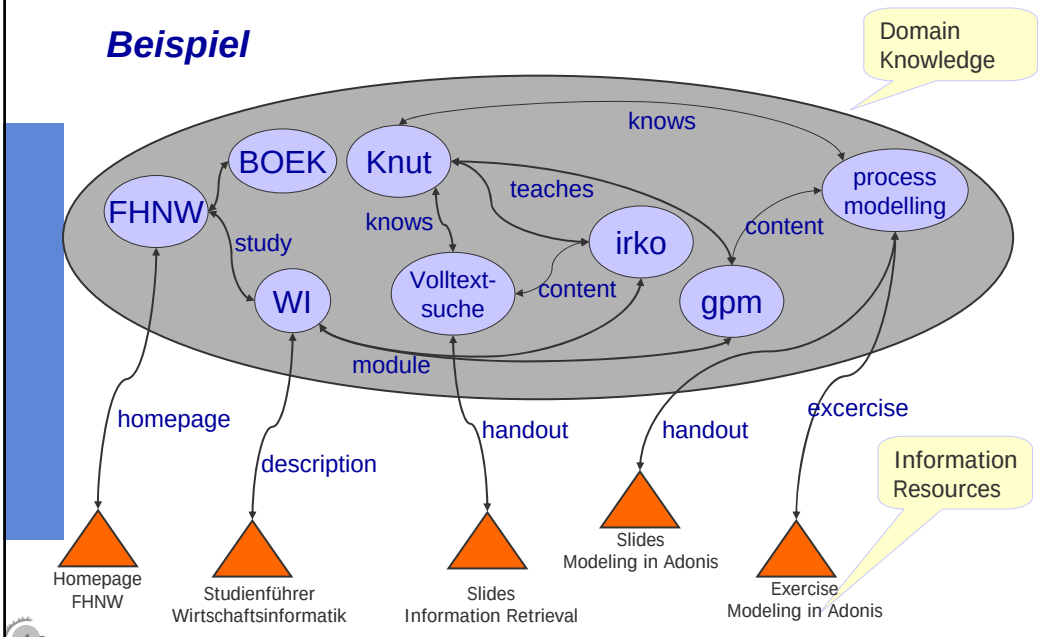
Topics
representing any concept like people, things, organizations, events

Associations
representing relationships between topics

Occurences
representing relationships between topics and information resources



Beispiel



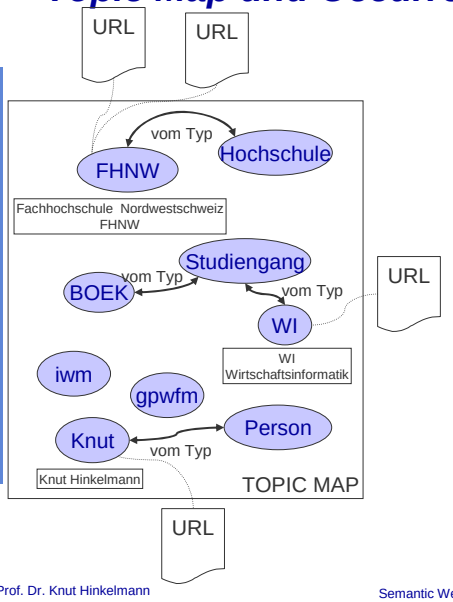
Topics, Subjects, Reification

- Topics repräsentieren elementare Subjekte im Kontext des modellierten Wissens
- Ein Subjekt kann alles sein, über das man sprechen oder nachdenken kann, z.B.
  - Person, Ausspruch, Land, Gefühl, Gegenstand, Wort, Zahl, Beziehung, ...
- Um in Topic Maps Aussagen über Subjekte machen zu können, muss man zu einem Subjekt ein Topic erstellen. Diesen Vorgang bezeichnet man als Reification
- Implizite Topics: Topics müssen in Dokumenten nicht explizit erwähnt werden, um trotzdem darin vorkommen zu können
  - Beispiel: Eine Gesetzesnovelle zum Steuerrecht wird den Begriff „Steuererhöhung“ nicht enthalten, obwohl er die Semantik der Novelle charakterisieren würde.

Topic Map Standardisierung

- ISO 13250 Topic Maps (HyTime)
  - Der Internationale Standard ISO/IEC 13250 definiert eine Standardnotation für den Austausch von Informationen über Topics and Relationen (auf Basis von SGML)
  - siehe http://www.ornl.gov/sgml/sc34/document/0058.htm
- **XML Topic Maps (XTM)**
  - XTM ist ein Produkt der TopicMaps.org, einem unabhängigen Konsortium
  - XTM soll kompatibel sein mit XML und ISO 13250
  - siehe http://www.topicmaps.org/xtm/1.0
- XML Schema for ISO 13250 Topic Maps
  - siehe http://www.diffuse.org/TopicMaps/schema.html

Topic Map und Occurrences



- Occurrences verbinden Topics mit externen Web-Ressourcen oder Dokumenten
- Ein Topic kann viele Occurrences aufweisen
- Jede Occurrence kann einen Type habe (instanceOf)
- Occurrence Typen sind wiederum Topics, die zuvor deklariert werden müssen

Topic Maps

```
<topicMap  id="fhnw-topicmap"
     xmlns="http://www.topicmaps.org/xtm/1.0/"

xmlns:xlink="http://www.w3.org/1999/xlink">

<topic>  …
</topic>

<topic> ...
</topic>

<association>  …
</association>

<mergeMap>  …
</mergeMap>

</topicMap>
```

- Eine Topic Map ist eine Sammlung von Topic Map Knoten: Topics, Associations und Scopes
- `<topicMap>` ist das Wurzel-Element einer Topic Map und kann `<topic>`-, `<association>`- und `<mergeMap>`-Elemente enthalten

Topics

```
<topic id="FHNW">
  <instanceOf>
    <topicRef xlink:href="#hochschule"/>
  </instanceOf>
  <baseName>
    <baseNameString>
        FHNW
    </baseNameString>
  </baseName>
</topic>
```

- Jeder Topic hat mindestens einen Name
  - **Base Name:** der eigentliche Name
    - `<baseName>`
  - **Variant Name:** Alternative Form, z.B. für die Ausgabe oder zum Sortieren.
- Ein Topic kann eine beliebige Zahl von Typen haben
- Typen sind eingeschlossen in das Element `<instanceOf>`
- Jeder Typ ist selbst ein Topic
  - entweder in der gleichen Map

    ```
    <topicRef
      xlink:href="#hochschule"/>
    ```
  - oder in einem anderen Dokument, z.B.

    ```
    <topicRef xlink:href=
      "univ.xtm#hochschule"/>
    ```

Occurrences

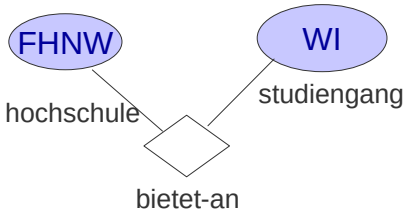
```
<topic id="FHNW">
   <baseName>
      <baseNameString> FHNW
      </baseNameString>
   </baseName>
<occurrence>
    <resourceData>
      Eine von 8 Schweizer FHs
    </resourceData>
 </occurrence>

<occurrence>
   <instanceOf>
     <topicRef xlink:href="#homepage"/>
   </instanceOf>
   <resourceRef xlink:href="http://www.fhnw.ch"/>
</occurrence>
</topic>
```

- Topics können Occurrences zugeordnet werden
- Zwei Arten von Occurrences:
  - Interne Ressource
    - String
    - Tag: `resourceData`
  - Externe Ressource:
    - Referenzen als URI
    - Tag: `resourceRef`
- Jede Occurrence kann einen Typ haben, der durch das Element `<instanceOf>` gegeben wird

Associations

```
<association>
  <instanceOf>
    <topicRef xlink:href="#bietet-an"/>
  </instanceOf>
  <member>
    <roleSpec>
        <topicRef xlink:href="#hochschule"/>
    </roleSpec>
    <topicRef xlink:href="#FHNW"/>
  </member>
  <member>
    <roleSpec>
        <topicRef xlink:href="#studiengang"/>
    </roleSpec>
    <topicRef xlink:href="#wi"/>
  </member>
</association>
```

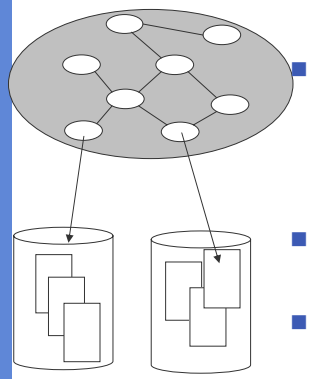


- Associations beschreiben Beziehungen zwischen Topics
- Associations sind nicht den Topics zugeordnet, sondern eigenständige Objekte
- Eine Assoziation kann maximal einen Typ haben, der wiederum ein Topic ist: `<instanceOf>` Beliebig viele Topics können an einer Relation teilhaben, die Topics sind Werte des Elements `member`
- Jedes Topic spielt dabei eine Rolle (Association Role)

Weitere Aspekte von Topic Maps

- Characteristics
  - Als Characteristics eine Topics bezeichnet man alles, was man über ein Topic aussagen kann, das sind
    - a topic name,
    - a topic occurrence, or
    - a role played by a topic as a member of an association
- Scopes
  - The assignment of such characteristics is considered to be valid within a certain scope, or context.
- Subject Indicator
  - Repräsentation des Subjects, das durch das Topic repräsentiert wird
  - Zwei Topics mit dem gleichen Subject Indicator sind identisch (d.h. sie "sprechen" über das gleiche Subjekt)

Anwendungen von Topic Maps



- Web Portals
  - Nutzen von Topic Maps zur Naviation im Portal: Aus Topic Maps werden HTML-Seiten generiert
- Knowledge-based Intranets
  - Semantisches Netz zur Repräsentation von Subjects der Organisation (wie Personen, Produkten, Rollen, Prozessen) und ihren Beziehungen sowie Links zu relevanten Dokumenten
- Content-Management Systems
  - Organisation von Inhalten mit einer Topic Map
- Enterprise Application Integration
  - Zusammenführen von Informationen aus verschiedenen Quellen: Getrennte Anwendungen erscheinen als Einheit

Outline of the Lecture

- Topic Maps
- F-Logic
- RDF/OWL

F-Logic: Definition von Klassenhierarchien und Relationen

- Subklassen-Beziehung

```
woman::person
man::person
```

- Definition von Relationen und Wertebereichen (range)

```
person[has_father=>man].
person[has_mother=>woman].

person[has_son=>>man].
person[has_daughter=>>woman].
```

=> Relation hat genau einen Wert
=>> Relationen hat 0, 1 oder mehrere Werte

F-Logic: Definition von Klassenhierarchien und Relationen

Klassen

```
woman::person
man::person
```

```
person[has_father=>man].
person[has_mother=>woman].

person[has_son=>>man].
person[has_daughter=>>woman].
```

Instanzen

```
abraham:man
isaac:man
sarah:woman
```

```
isaac[has_father->abraham].
isaac[has_mother->sarah].
sarah[has_son->>isaac].
abraham[has_son->>{isaac,ishmael}].
```

```
FORALL X,Y   X[has_son->>Y]   <- Y:man[has_father->X].
```

```
FORALL X,Y   <- X:woman[has_son->>Y[has_father->abraham]].
```

=> Relation hat genau einen Wert
=>> Relationen hat 0, 1 oder mehrere Werte

F-Logic: Definition von Instanzen

- Zuordnung von Instanzen zu Klassen (entspricht rdf:type)

```
abraham:man
isaac:man
sarah:woman
```

- Zuordnung von Attributwerten

```
isaac[has_father->abraham].
isaac[has_mother->sarah].
sarah[has_son->>isaac].
abraham[has_son->>{isaac,ishmael}].
```

Werden mehrere Werte angegeben, müssen diese in geschweifte Klammern gesetzt werden.

- Kombination von Klassenzuordnung und Attributwerten

```
isaac:man[has_father->abraham:man].
isaac[has_mother->sarah:woman].
```

-> Relation hat genau einen Wert
->> Relationen hat 0, 1 oder mehrere Werte

F-Logic: Abkürzende Schreibweisen

- Relationen zu einem Objekt können zu einem Ausdruck zusammengefasst werden:

```
jacob:man[has_father->isaac;
          has_son->>{joseph:man,
                     benjamin:man[has_mother->rahel]}].
```

ist äquivalent zu:

```
jacob:man.
jacob[has_father->isaac].
jacob[has_son->>joseph].
joseph:man.
jacob[has_son->>benjamin:man].
benjamin[has_mother->rahel].
```

F-Logic: Regeln

- Regeln bestehen aus Kopf und Rumpf (Bedingungen)
  - Kopf und Rumpf werden durch das Zeichen <- getrennt
  - Alle Variablen müssen durch FORALL deklariert sein
- Beispiel:

```
FORALL X,Y   X[has_son->>Y]  <- Y:man[has_father->X].
```

Variablen *Regelkopf* *Bedingung*

"Für alle X und Y gilt: X hat einen Sohn Y, wenn Y ein Mann ist und den Vater X hat."

F-Logic: Regeln

- Beispiele:

```
FORALL X,Y   X[has_son->>Y]  <- Y:man[has_father->X].
FORALL X,Y   X[has_son->>Y]  <- Y:man[has_mother->X].
FORALL X,Y   X[has_daugther->>Y]  <- Y:woman[has_father->X].
FORALL X,Y   X[has_daugther->>Y]  <- Y:woman[has_mother->X].
```

```
FORALL X,Y   X[ancestor->>Y] <- X[father->Y].
FORALL X,Y   X[ancestor->>Y] <- X[mother->Y].
FORALL X,Y,Z   X[ancestor->>Y] <- X[father->Z] AND Z[ancestor->>Y].
FORALL X,Y,Z   X[ancestor->>Y] <- X[mother->Z] AND Z[ancestor->>Y].
```

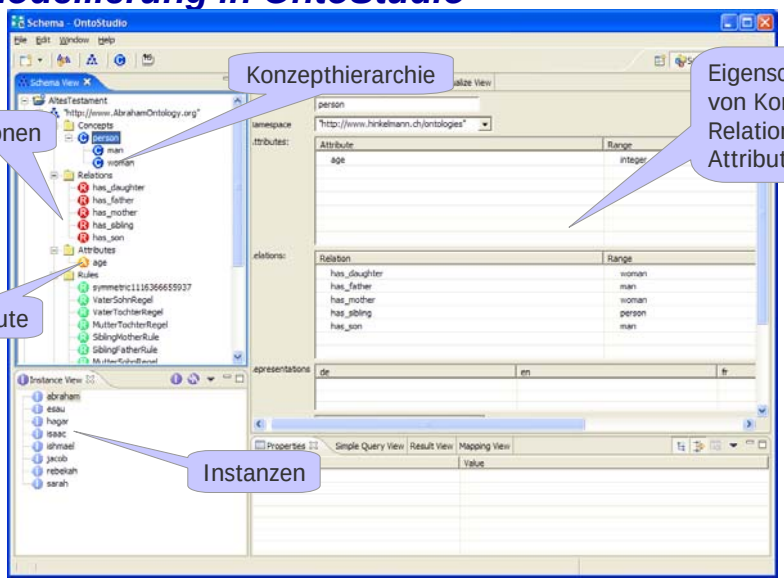
Anfragen

- Anfragen sind Regeln ohne Kopf
- Beispiel:

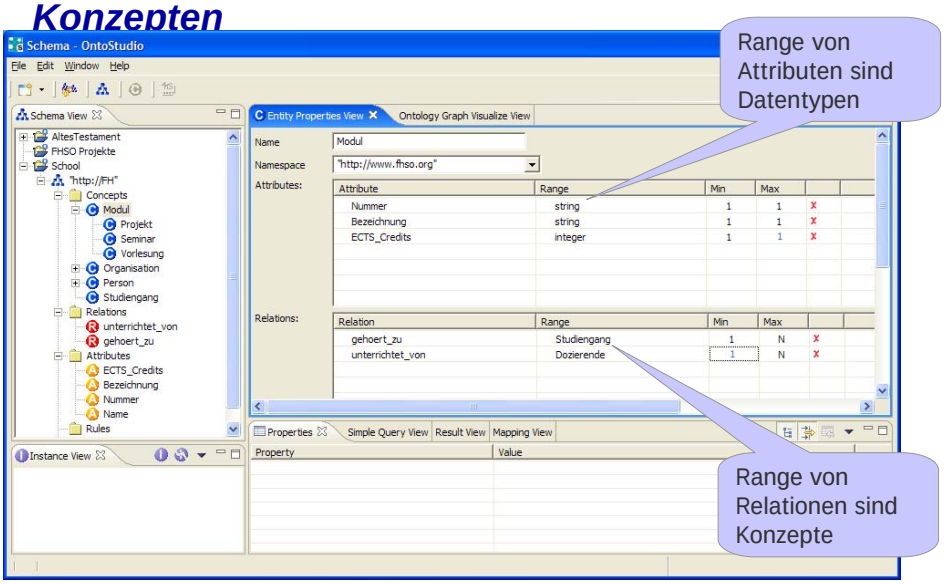
```
FORALL X,Y   <- X:woman[has_son->>Y[has_father->abraham]].
```

- Auswertungsstrategie für Anfrage: Backward Chaining wie in logischer Programmierung

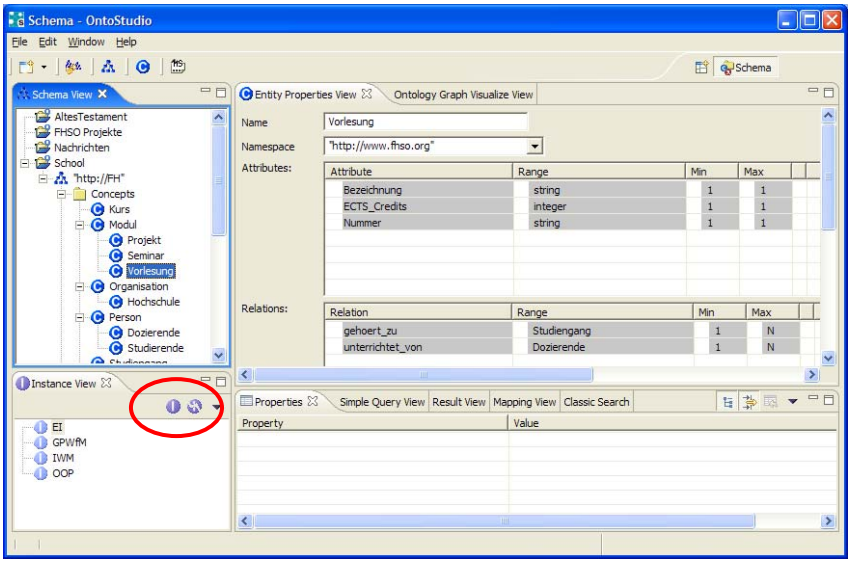
Modellierung in OntoStudio



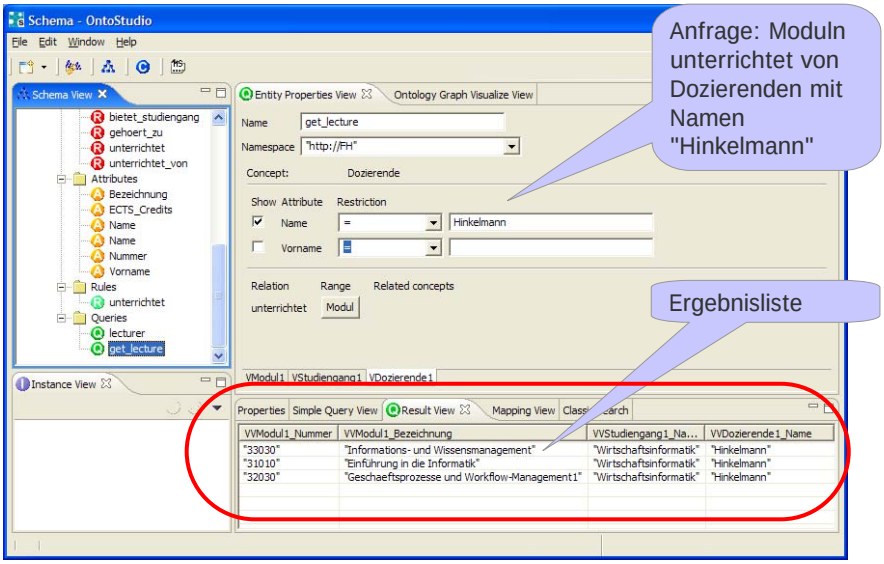
Definition von Attributen und Relation zu Konzepten



Instanzen anlegen



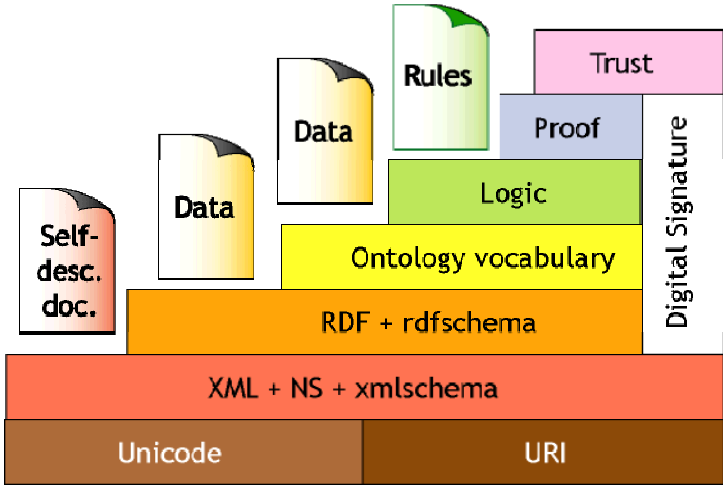
Definition von Anfragen



Outline of the Lecture

- Topic Maps
- F-Logic
- RDF/OWL

W3C Semantic Web - Architektur



Quelle: T. Berners-Lee, http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

W3C Standardisierungen zum Semantic Web

- XML (eXtensible Markup Language)
  - Strukturierte Dokumente zum Datenaustausch
  - Keine explizite Bedeutung der Dokumente und Inhalte
- RDF (Resource Description Framework)
  - Beschreibung von Web-Ressourcen
  - Datenmodell für Objekte (Ressource) und Beziehungen (Property) dazwischen
- RDF/S (RDF Schema)
  - Erweiterung von RDF zur Definition von Klassen, Beziehungen und Klassenhierarchien
- OWL (Web Ontology Language)
  - Erweiterung von RDF Schema zur Beschreibung von Klassen und Eigenschaften, z.B. Disjunktheit von Klassen, Kardinalität ("genau eine Instanz"), usw.

Literatur: http://www.w3.org

RDF – Resource Description Framework

RDF - Metadaten über Ressourcen

- Ressourcen in RDF
  - … sind Web-Ressourcen, aber auch beliebige Dinge innerhalb und ausserhalb des WWW
- *Metadaten in RDF*
  - … *sind maschinen-verständliche Informationen über Ressourcen*
  - … sind repräsentiert als Aussagen in Form von Tripeln: Subjekt-Prädikat-Wert
  - … entsprechen einem semantischen Netz

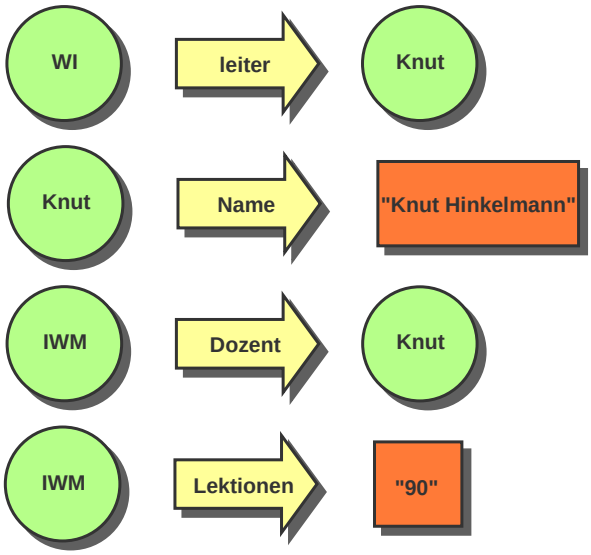
Grundbaustein von RDF: Das Tripel

Ressource
- Als Ressource bezeichnet man alle Dinge, die durch RDF-Ausdrücke beschrieben werden (Subjekte).
- Ressourcen können sein: Webseiten, Teile von Webseiten, Elemente von XML-Dokumenten oder auch Dinge, die nicht direkt über das Web zugreifbar sind (z.B. Bücher, Personen, Fahrzeuge, ...).

Literal
- Literale sind atomare Werte (Daten) wie Unicode-Zeichenketten

Property
- Eigenschaften von Ressourcen (Prädikate)
- Werte von Properties sind Ressourcen oder Literale
  (Properties, deren Werte eine Ressource ist, nennt man auch Relationen)

Grundbaustein von RDF: Das Tripel - Beispiele



URI in RDF

- Ziel: eindeutiger Identifier für Ressourcen und Beziehungen
- Grundprinzip: **Alles kann durch eine URI bezeichnet werden**
- URI stehen für
  - Ressourcen im WWW
  - externe Ressourcen
  - Properties
- Ein URI beginnt mit einem Protokoll-Bezeichner (z.B. http:, ftp:, mailto:) gefolgt von einer Adresse
- Idee: Wenn man für Objekte eine URI mit dem eigenen Domainnamen verwendet (eindeutig), vermeidet man Namenskonflikte mit Objekten anderer Benutzern

URI – Beispiele

```
http://www.ietf.org/rfc/rfc791.txt
http://www.w3.org/People/Berners-Lee
mailto:knut.hinkelmann@fhnw.ch
urn:isbn:0451450523
http://www.xmlns.com/foaf/0.1/Person
http://www.xmlns.com/foaf/0.1/knows
http://www.w3.org/2000/01/rdf-schema#Class
#iwm-skript
```

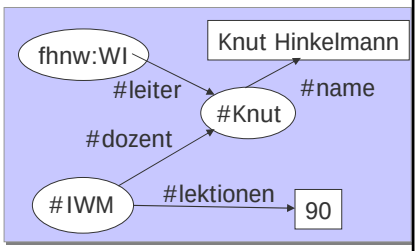
Eine URI mit einem "#" referenziert ein Element innerhalb eines Dokuments:

`http://www.w3.org/2000/01/rdf-schema#Class` meint das Element **`Class`** in der Adresse **`http://www.w3.org/2000/01/rdf-schema`**

`#iwm-skript` meint das Element **`iwm-skript`** im aktuellen Dokument

Notationen für RDF Statements

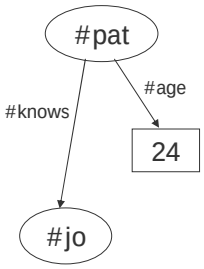
- Graphisch (Semantisches Netz)
  - ovale Knoten sind Ressourcen
  - eckige Knoten sind Literale
  - Kanten sind Properties
- Tripel
  - N3 besteht aus einer Menge von Subjekt-Prädikat-Wert Tripeln
    - das Subjekt ist eine Ressource
    - das Prädikat ist eine Property
    - der Wert ist eine Ressource oder Literal
- XML
  - W3C Standard
  - automatische Verarbeitung möglich



Subjekt	*Prädikat*	*Wert*
fhnw:WI	<#leiter>	<#Knut>
<#IWM>	<#dozent>	<#Knut>
<#IWM>	<#lektionen>	"90"
<#Knut>	<#name>	"Knut Hinkelmann"

Die Notation 3 (N3) für RDF

Subjekt, Prädikat und Objekt



- In N3 werden Aussagen als Tripel geschrieben und mit einem Punkt abgeschlossen:

  <#pat> <#knows> <#jo> .

  - <#pat> ist das Subjekt: Eine Ressource, über die eine Aussage gemacht wird
  - <#knows> ist eine Eigenschaft (Property)
  - <#jo> ist der Wert der Eigenschaft für das Subjekt
- Werte sind entweder
  - Ressourcen – identifiziert durch eine URI

    <#pat> <#knows> <#jo> .
  - oder Literale - Strings oder Zahlen

    <#pat> <#age> "24" .

siehe: http://www.w3.org/2000/10/swap/Primer.html

Notation N3 für RDF

URIs und Literale

- Auch in N3 werden alle Ressourcen und Eigenschaften durch eine URI identifiziert, z.B.

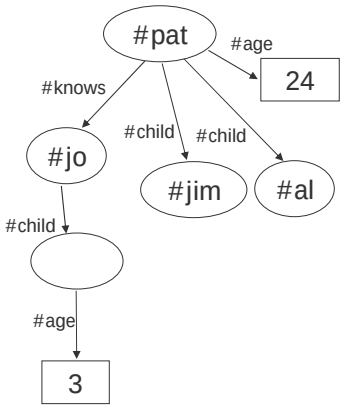
  `<http://www.fhnw.ch/wi/knut>`
- Das Zeichen # ist eine Abkürzung für die URI des aktuellen Dokuments, z.B.

  `<#knut>`
- Das aktuelle Dokument wird mit <> bezeichnet

  `<> <#title> "Semantic Web" .`
- Lediglich Literale (Strings oder Zahlen) sind keine URIs, z.B.

  `<#knut> <#name> "Knut Hinkelmann" .`

N3: Implizite Ressourcen und Bezeichner



- Semikolon ";" trennt mehrere Eigenschaften zur gleichen Ressource

```
<#pat> <#knows> <#jo>; <#age> "24"
```

- Komma "," trennt mehrere Werte der gleichen Property

```
<#pat> <#child> <#al>, <#jim>.
```

- Semikolon und Komma können kombiniert werden

```
<#pat> <#child> <#al>, <#jim>;
       <#knows> <#jo>;
       <#age> "24".
```

- Eckige Klammern "[" und "]" beschreiben Ressourcen, ohne sie explizit anzugeben, z.B.

```
<#jo> <#child> [ <#age> "3"].
```

soll ausdrücken, dass Jo ein Kind im Alter von 3 Jahren hat

Bezeichner

- Identifier sind lediglich Bezeichner und für die Maschine ohne Semantik
  - Wir können statt <#pat> oder <#child> auch <#xyz> schreiben

Die Bedeutung ergibt sich durch die Eigenschaften.

- Dass <#pat> für eine Person mit Namen "Pat" steht muss explizit ausgedrückt werden z.B.

```
<#pat> <#name> "Pat" .
```

- Bei der Definition von Konzepten kann man bereits existierende Definitionen wiederverwenden. Dies geschieht durch Verwendung der exakt gleichen URI
  - Beispiel: Das Konzept "title" wurde von einer Gruppe Dublin Core unter dem Identifier

```
<http://purl.org/dc/elements/1.1/title>
```

definiert. Dieses Konzept kann man wiederverwenden:

```
<> <http://purl.org/dc/elements/1.1/title> "My family" .
```

Namespaces/Prefixes in N3

- Namespaces erlauben eine kompaktere Darstellung von Ressourcen und Properties
- In N3 werden die Namespaces durch "`@prefix`" gesetzt

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
<> dc:title  "My family" .
```

- Beachte: Bei der Verwendung von Präfixes
  - entfallen die Klammern < und >
  - steht ein "`:`" statt eines "#"
- Der leere Präfix steht für das aktuelle Dokument

```
@prefix : <#> .
```

wodurch das obige Beispiel wie folgt geschrieben werden kann:

```
:pat :child [ :age "4"] , [ :age "3"] .
```


RDF-XML: Eine XML-basierte Syntax für RDF

Struktur eines RDF-XML-Dokuments

- Ein RDF-XML-Dokument enthalt ein Element rdf:RDF mit mehreren Description-Elementen
- Ein Description-Element fasst die Properties einer Ressource zusammen
  - Ressourcen als Werte werden als Attributwerte dargestellt
  - Literale als Werte sind Elemente

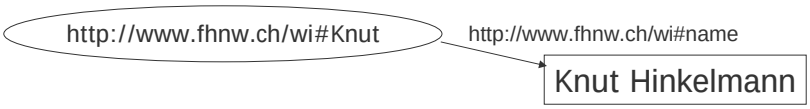
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         weitere Namespaces >

  <rdf:Description rdf:about="Resource">
     <Property1 rdf:resource="Resource2"/>
     <Property2> Literal < Property2>
   </rdf:Description>
...
</rdf:RDF>
```

Property2 ist eine Eigenschaft mit einem Literal als Wert

Der Wert von Property1 ist eine Ressource, angegeben als Wert des Attributs rdf:resource (für Resource2 gibt es wieder ein Description-Element)

Beispiel eines RDF-XML Statements



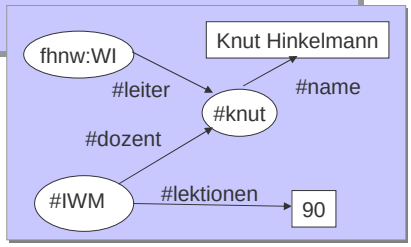

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:wi="http://www.fhnw.ch/wi#">
  <rdf:Description rdf:about="http://www.fhnw.ch/wi#Knut">
      <wi:Name> Knut Hinkelmann </wi:Name>
  </rdf:Description>
</rdf:RDF>
```

- Die **Ressource** ist der Wert des Attributs rdf:about
- die **Property** ist der Tag des Elements innerhalb von rdf:Description
- Das **Literal** ist der Inhalt des Property-Tags
- "wi" ist ein Namespace

RDF-XML: Beispiel

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:wi="http://www.fhnw.ch/wi#">
  <rdf:Description rdf:about="http://www.fhnw.ch/wi">
      <wi:leiter rdf:resource="http://www.fhnw.ch/wi#Knut"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.fhnw.ch/wi#Knut">
    <wi:name> Knut Hinkelmann</wi:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.fhnw.ch/wi#IWM">
    <wi:dozent rdf:resource="http://www.fhnw.ch/wi#Knut"/>
    <wi:lektionen> 90 </wi:lektionen>
  </rdf:Description>
</rdf:RDF>
```

3 Description-Elemente, da es drei Ressourcen mit Properties gibt



Unter http://www.w3.org/RDF/Validator kann man RDF-Dokumente prüfen und als Graph darstellen

rdf:about und rdf:ID

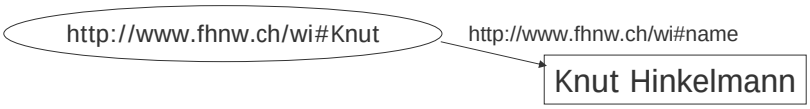
- Der Wert des Arguments `rdf:about` ist eine vollständige URI
- Der Wert von `rdf:ID` ist ein Identifier, der im aktuellen Dokument eindeutig ist

```
<rdf:Description rdf:ID="pat">
   <s:age> 24 </s:age>
</rdf:Description>
```

- Die vollständige URI erhält man durch
  - die Basis-URI des Dokuments
  - gefolgt von einem #
  - und dem Wert von `rdf:ID`
- Auf den Namen kann man durch Anfügen an # verweisen, z.B. `#pat`

```
<rdf:Description rdf:ID="jo">
   <s:knows rdf:resource="#pat"/>
</rdf:Description>
```


Abkürzende Darstellungen in RDF/XML



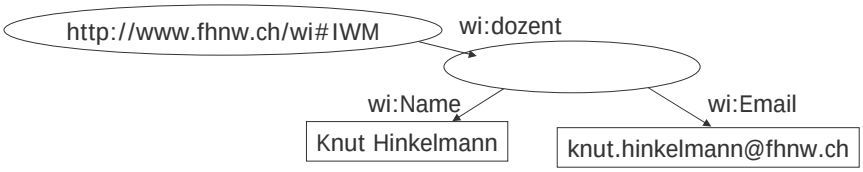
Properties können auch als Attribute von Description statt als Elemente dargestellt werden.

Beispiel:

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:wi="http://www.fhnw.ch/wi#">
  <rdf:Description rdf:about="http://www.fhnw.ch/wi#Knut"
                   wi:Name="Knut Hinkelmann" />
</rdf:RDF>
```

Implizite Ressourcen in RDF/XML

"Der Dozent von IWM ist jemand mit Namen "Knut Hinkelmann" und Email knutl.hinkelmann@fhnw.ch"



entspricht in RDF/XML dem Element

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:wi="http://www.fhnw.ch/wi#">
  <rdf:Description rdf:about="http://www.fhnw.ch/wi#IWM">
    <wi:dozent wi:Name="Knut Hinkelmann"
               wi:Email="knut.hinkelmann@fhnw.ch"/>
</rdf:Description>
</rdf:RDF>
```

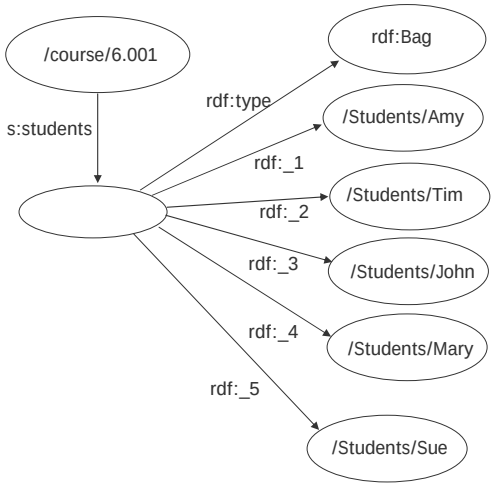

Container in RDF

Container entsprechen Datenstrukturen, mit denen für eine Property mehrere Ressourcen als Werte definiert werden können:

- **Bag:** eine ungeordnete Liste von Ressourcen oder Literalen, Werte können mehrfach auftreten (rdf:Bag)
  - Beispiel: Menge der Teilnehmer eines Kurses
- **Sequence:** eine geordnete Liste von Ressourcen oder Literalen (rdf:Seq)
  - Beispiel: Alphabetische Liste der Kursteilnehmer
- **Alternative:** Eine Liste von Ressourcen oder Literalen, die Alternativen für einen Wert einer Eigenschaft darstellen (rdf:Alt)
  - Beispiel: Titel des Kurses in mehreren Sprachen

Beispiel: Einfacher Bag Container

"Die Studierenden in Kurs 6.001 sind Amy, Tim, John, Mary und Sue"



- Der Wert der Property s:students is ein Bag
- In der graphischen Darstellung werden die Bag-Elemente durch-nummeriert durch das Suffix "_1", "_2" usw.

Beispiel: Einfacher Bag Container in RDF/XML

"Die Studierenden in Kurs 6.001 sind Amy, Tim, John, Mary und Sue"

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:s="http://example.org/schema/">
 <rdf:Description rdf:about="http://mycollege.edu/courses/6.001">
   <s:students>
     <rdf:Bag>
       <rdf:li resource="http://mycollege.edu/students/Amy"/>
       <rdf:li resource="http://mycollege.edu/students/Tim"/>
       <rdf:li resource="http://mycollege.edu/students/John"/>
       <rdf:li resource="http://mycollege.edu/students/Mary"/>
       <rdf:li resource="http://mycollege.edu/students/Sue"/>
     </rdf:Bag>
   </s:students>
 </rdf:Description>
</rdf:RDF>
```

Der Wert der Property s:students ist ein Bag. Jedes Element eines Bag wird durch das Element rdf:li angegeben (vgl. Listenelemente in HTML)

Aussage über Container und Elemente von Containern

```
<rdf:Bag ID="course">
  <rdf:li resource="http://foo.org/DB.html" />
  <rdf:li resource="http://bar.org/OOP.html" />
</rdf:Bag>

<rdf:Description rdf:about="#course">
  <s:responsible>Peter Miller</s:responsible>
</rdf:Description>
```

- Das obige Beispiel besagt, dass Peter Miller für den Kurs bestehend aus den Moduln DB und OOP verantwortlich ist. Es wird nichts über die einzelnen Module ausgesagt.
- Um auszudrücken, dass Peter Miller für jedes der Module verantwortlich ist, gibt es das Attribut abouteach

```
<rdf:Description rdf:abouteach="#course">
  <s:responsible>Peter Miller</s:responsible>
</rdf:Description>
```


Aussage über alle Elemente mit gleichem Präfix

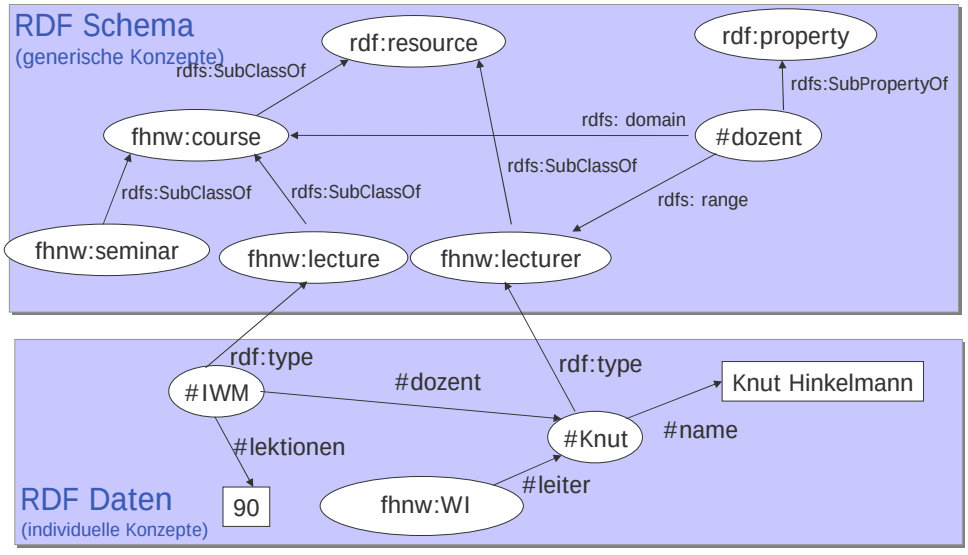
- Oft kann oder will man nicht alle Elemente eines Bags auflisten, z.B. alle Elemente einer Website
- Mit dem Attribut aboutEachPrefix kann man aussagen über alle Elemente mit gleichen URI-Präfix machen
- Das folgende Element beschreibt die Eigenschaft Copyright für jede Seite unterhalb von http://www.fhnw.ch/

```
<rdf:Description rdf:aboutEachPrefix="http://www.fhnw.ch/">
  <s:Copyright>
    © 2003 Fachhochschule Nordwestschweiz
  </s:Copyright>
</rdf:Description>
```

RDF Schema

- RDF ermöglicht einfache Aussagen über Ressourcen, Properties und Werte
- RDF Schema ist eine Erweiterung von RDF
- RDF Schema stellt Mechanismen zur Vorfügung, um ein Vokabular zu definieren:
  - Definition von Klassen (rdfs:Class)
  - Anordnung von Klassen in einer Taxonomie (rdfs:SubClassOf)
  - Zuordnung von Instanzen zu Klassen (rdf:Type)
  - Definition von Properties (rdfs:Property)
  - Anordnung von Properties in einer Taxonomie (rdfs:SubPropertyOf)
  - Einschränkung von Werten für Properties (rdfs:domain, rdfs:range)
- RDF Schema ist selbst wiederum in RDF beschrieben
- Das Kernvokabular von RDF Schema ist im Namespace rdfs definiert, dessen URI ist http://www.w3.org/2000/01/rdf-schema#

RDF und RDF Schema



RDF-Schemadefinitionen

- Durch die Property `rdf:type` wird eine Instanz einer Klasse zugeordnet

```
:Pat    rdf:type    :Person .          (Pat ist eine Person)
```

- Eine Ressource wird zu einer Klasse, indem man sie mit `rdf:type` der vordefinierten Klasse `rdfs:Class` zuordnet:

```
:Person  rdf:type  rdfs:Class .        (Person ist eine Klasse)
```

- Mit `rdfs:subClassOf` wird einer Klasse eine Superklasse zugordnet

```
:Woman rdf:type        rdfs:Class;     (Woman ist eine Klasse)
       rdfs:subClassOf :Person .       (Woman ist Subklasse von Person)
```

- Um zu einer Property Eigenschaften definieren zu können, macht man sie sie zu einer Ressource (→ Reification) indem man sie der Klasse `rdf:Property` zuordnet:

```
:sibling  rdf:type  rdf:Property .     (Sibling ist eine Property)
:sister   rdf:type  rdf:Property .     (Sister ist eine Property)
```

- Mit `rdfs:SubPropertyOf` kann man eine Hierarchie von Eigenschfaten definieren

```
:sister  rdfs:SubPropertyOf  :sibling;  (Alle Schwestern sind auch
                                         Geschwister)
```

- Definitions- und Wertebereich für Properties

```
:sister  rdfs:domain  :Person;      (Sister ist eine Eigenschaft von Person,
         rdfs:range   :Woman .      die Werte sind aus der Klasse Woman)
```

Vordefinierte Klassen in RDF Schema

- Alle Datentypen in RDF sind Klassen
- Folgende Klassen sind vordefiniert:
  - **`rdfs:Class`**
    Alle Klassen sind Instanzen dieser Klasse
  - **`rdfs:Resource`**
    Die allgemeinste Klasse, alle anderen Klassen sind Subklassen dieser Klasse
  - **`rdfs:Literal`**
    Die Klasse aller Literalen Werte, eine Subklasse von rdfs:Resource
  - **`rdfs:Datatype`**
    Klasse von Datentypen wie Integer, Boolean usw., jede Instanz von rdfs:Datatype ist eine Subklasse von rdfs:Literal
  - **`rdf:XMLLiteral`**
    Eine Subklasse von rdfs:Literal, zum Einbinden von XML in RDF
  - **`rdf:Property`**
    Die Klasse aller Properties

Vordefinierte Properties in RDF Schema

`rdf:type`
Ordnet eine Instanz einer Klasse zu

`rdfs:SubClassOf`
Subklassen-Beziehung: Eine Klasse K ist Subklasse einer anderen Klassen K' genau dann, wenn alle Instanzen von K auch Instanzen von K' sind

`rdfs:range`
Ordnet einer Property eine Klasse möglicher Werte zu (Wertebereich)

`rdfs:domain`
Ordnet einer Property eine Klasse von Ressoucen zu, die diese Propetry haben können (Definitionsbereich)

`rdfs:SubPropertyOf`
Eine Property P ist Subproperty einer anderen Property P' genau dann, wenn alle Ressourcen, die in Eigenschaft P zueinander stehen, auch in Eigenschaft P' zueinander stehen

`rdfs:label`
für den Menschen verständlicher Bezeichner

`rdfs:comment`

Beschreibung von Properties

- In RDF Schema kann man Properties als Instanzen der Klasse rdfs:Properties definieren und ihnen dann Eigenschaften zuordnen
- Beispiel:

```
:Person   rdf:type     rdfs:Class
:Book     rdf:type     rdfs:Class
:author   rdf:type     rdf:Property
:author   rdfs:range   :Person
:author   rdfs:domain  :Book
```

`:Person` und `:Book` sind Klassen, `:author` ist eine Property. Aussagen mit der Eigenschaft `:author` haben Instanzen von `:Person` als Objekte und Instanzen von `:Book` als Subjekte

- Beachte: Properties müssen als Ressourcen definiert sein (→ Reification)!