

# **FH Wedel Informatik Seminar WS2006/2007 : Service-orientierte Architektur (SOA)**

*Christian Köhn 13.12.2006*

---

**OWL: Zweck, Aufbau und Beispiel**

## **Die Beschreibungssprache OWL**

**Christian Köhn :: mi2219  
Informatik Seminar WS2006/2007**

# Inhaltsübersicht

- Einleitung
  - Motivation
  - Ontologien
- Web Ontology Language OWL
  - Entstehungsgeschichte
  - Ausprägungen
  - Aufbau
  - Sprachausdrücke und Beispiel
  - Tools
  - Reasoning

# Einleitung

# Einleitung

## Motivation

Das Internet stellt uns heute eine Fülle von Informationen bereit. Diese Informationen sind auf zig milliarden Seiten verteilt. Doch wie findet ein User/Computer relevante Informationen?

Heutige Suchmaschinen werden gesteuert durch:

- Stichworte
- Muster (sowohl bei Text als auch bei Bildern)

Durch diese rein auf der Syntax beruhende Suche bleiben viele relevante Informationen verborgen.  
Zum Beispiel durch die Wahl eines falschen/ungenauen Stichworts

# Einleitung

## Motivation

Das Internet stellt uns heute eine Fülle von Informationen bereit. Diese Informationen sind auf zig milliarden Seiten verteilt. Doch wie findet ein User/Computer relevante Informationen?

Heutige Suchmaschinen werden gesteuert durch:

- Stichworte
- Muster (sowohl bei Text als auch bei Bildern)

Durch diese rein auf der Syntax beruhende Suche bleiben viele relevante Informationen verborgen.  
Zum Beispiel durch die Wahl eines falschen/ungenauen Stichworts

Ziel: Suchanfragen aufgrund ihrer Bedeutung bearbeiten zu können (Verstehen vs Textmatching)

# Einleitung

## Beispiel

Welche Informationen sehen wir?

Lehrveranstaltungen von [Prof. Dr. Sebastian Iwanowski](#) an der [FH Wedel](#) im WS 2006/2007:

### Pilotvorlesung am OE-Tag

Studiengänge: alle

Di, 10.10.2006, 16:00 Uhr - 17:00 Uhr, HS2

### Vorlesung Diskrete Mathematik

Studiengänge: B\_Infl, B\_TInfl, B\_MInfl, B\_WInfl, II6, WI6, MS1

Di, Mi 14:00 Uhr - 15:15 Uhr, HS 2, Große Übung (bei [Maximilian Herold](#)) Mo 14:00 Uhr - 15:15 Uhr, HS 2

### Vorlesung Grundlagen der Theoretischen Informatik

Studiengänge: B\_Infl, B\_TInfl, B\_MInfl, B\_WInfl

Do 08:00 Uhr - 09:15 Uhr, HS 6, Große Übung (bei [Maximilian Herold](#)) Mo 11:00 Uhr - 12:15 Uhr, HS 6

### Vorlesung Software-Engineering

Studiengänge: B\_Infl2, B\_TInf45, B\_MInf3, B\_WInf12, IA5, AI1

Do 17:00 Uhr - 18:15 Uhr, HS 2

### Vorlesung Rechnernetze

Studiengänge: B\_Inf34, B\_TInf3

Mi, Do 15:30 - 16:45 Uhr, HS 4

### Vorlesung Wissensbasierte Systeme

Studiengänge: B\_Inf45, B\_TInf45, II6, WI6, MS1

Mi 8:00 Uhr - 9:15 Uhr, HS 4

# Einleitung

## Beispiel

Welche Informationen sehen wir?

Syntax:

- HTML Markup (Schriftgröße, Farbe ...)
- Links zu anderen Seiten

Semantik:

- Menschen verstehen die Semantik der Seite relativ einfach.
- Computer hingegen können die Bedeutung der Inhalte nur sehr schwer erschliessen.

Lehrveranstaltungen von [Prof. Dr. Sebastian Iwanowski](#) an der [FH Wedel](#) im WS 2006/2007:

### [Pilotvorlesung am OE-Tag](#)

Studiengänge: alle  
Di, 10.10.2006, 16:00 Uhr - 17:00 Uhr, HS2

### [Vorlesung Diskrete Mathematik](#)

Studiengänge: B\_Infl, B\_TInfl, B\_MInfl, B\_WInfl, II6, WI6, MS1  
Di, Mi 14:00 Uhr - 15:15 Uhr, HS 2, Große Übung (bei [Maximilian Herold](#)) Mo 14:00 Uhr - 15:15 Uhr, HS 2

### [Vorlesung Grundlagen der Theoretischen Informatik](#)

Studiengänge: B\_Infl, B\_TInfl, B\_MInfl, B\_WInfl  
Do 08:00 Uhr - 09:15 Uhr, HS 6, Große Übung (bei [Maximilian Herold](#)) Mo 11:00 Uhr - 12:15 Uhr, HS 6

### [Vorlesung Software-Engineering](#)

Studiengänge: B\_Infl2, B\_TInf45, B\_MInf3, B\_WInf12, IA5, AI1  
Do 17:00 Uhr - 18:15 Uhr, HS 2

### [Vorlesung Rechnernetze](#)

Studiengänge: B\_Inf34, B\_TInf3  
Mi, Do 15:30 - 16:45 Uhr, HS 4

### [Vorlesung Wissensbasierte Systeme](#)

Studiengänge: B\_Inf45, B\_TInf45, II6, WI6, MS1  
Mi 8:00 Uhr - 9:15 Uhr, HS 4

# Einleitung

## Beispiel

Welche Informationen sieht ein Computer?



# Einleitung

## Beispiel

Stellt XML in diesem Zusammenhang eine Lösung dar?

- Eine Änderung der XML Bezeichner erfordert auch eine Änderung der Software
- Stellt nur ein Abkommen zwischen zwei/mehreren Instanzen dar
- Ist somit keine allgemeingültige formale Definition

```
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ?????????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????? ????????????????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
```

# Einleitung

## Beispiel

Stellt XML in diesem Zusammenhang eine Lösung dar?

- Eine Änderung der XML Bezeichner erfordert auch eine Änderung der Software
- Stellt nur ein Abkommen zwischen zwei/mehreren Instanzen dar
- Ist somit keine allgemeingültige formale Definition

```
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ?????????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????????????????? ????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
<vorlesung>????????????????</vorlesung>
<hoerer>?????? ?????? ????????????????????????????? ?????</hoerer>
<zeit>????????????????????</zeit>
<uebung>???????????????????? ????????????????? ?????????</uebung>
```

Lösung: Ontologien, um die Bedeutung und Beziehungen von Begriffen formal zu beschreiben.

# Was ist eine Ontologie?

Ontologien befassen sich mit der formalen Repräsentation von Wissen.

Unter einer Ontologie versteht man in der Informatik im Bereich der Wissensrepräsentation ein formal definiertes System von Konzepten und Relationen.

Wikipedia / Ontologie (Informatik)

Ontologie ist ein aus der Philosophie entlehener Ausdruck, der sich auf die Wissenschaft bezieht, die die unterschiedlichen Arten von Entitäten in der Welt und ihre Beziehungen untereinander beschreibt.

w3c OWL Web Ontology Language Guide

Eine Ontologie ist eine explizite formale Spezifikation einer gemeinsamen Konzeptualisierung

T. R. Gruber: A translation approach to portable ontologies. In: Knowledge Acquisition, Band 5, Nummer 2, Seite 199-220, 1993

# Was ist eine Ontologie?

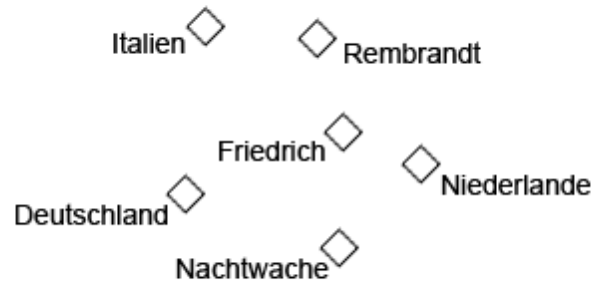
Ontologien setzen sich aus folgenden Bestandteilen zusammen:

- Instanzen/Individuen
- Relationen/Eigenschaften
- Konzepte
- Einschränkungen von Eigenschaften
- Axiome

# Was ist eine Ontologie?

## Instanzen/Individuen

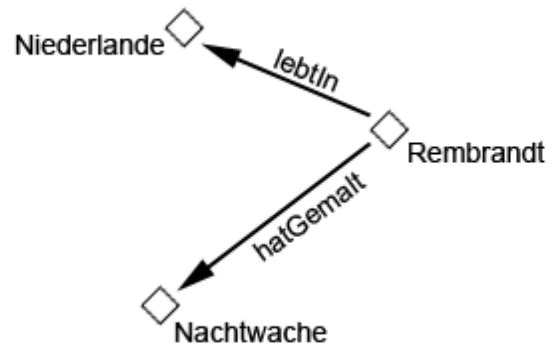
Individuen repräsentieren Objekte innerhalb des Wissensbereiches an dem wir interessiert sind.



# Was ist eine Ontologie?

## Eigenschaften

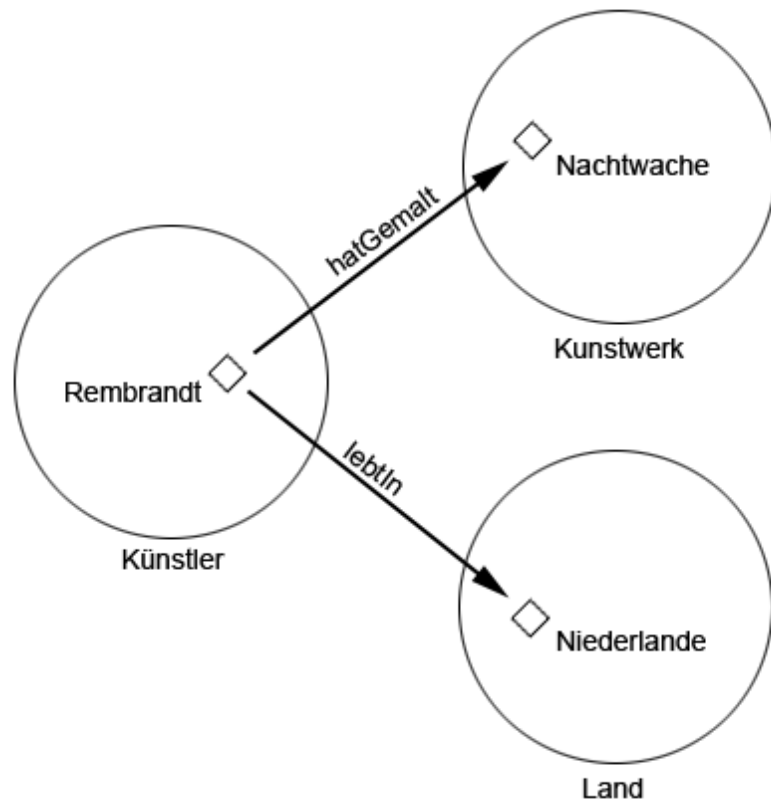
Eigenschaften definieren allgemeingültige Fakten über Klassenmitglieder und spezifische Fakten über Individuen. Sie stellen hierbei eine binäre Relation dar.



# Was ist eine Ontologie?

## Konzepte

Konzepte werden auch als Klassen bezeichnet. Sie beschreiben gemeinsame Eigenschaften von Dingen.



# Was ist eine Ontologie?

## Nutzen

Ontologien sind wichtig um ein *gemeinsames* Verständnis von Informationen zu bekommen.

Hierbei spielt es keine Rolle mehr, welcher Art die Kommunikationspartner angehören:

- Mensch <-> Mensch
- Software-Agent <-> Software-Agent
- Mensch <-> Software

Durch logisches Folgern (Inferenz) lassen sie so folgende Aussagen machen:

- Rückschlüsse aus den vorhanden Daten ziehen
- Widersprüche erkennen
- Fehlendes Wissen aus dem vorhanden selbstständig ergänzen

Ontologien erlauben ausserdem die Wiederverwendung/Erweiterung bestehender Ontologien.

# Web Ontology Language (OWL)



# Web Ontology Language (OWL)

## Entstehungsgeschichte

- RDF/RDFS war die erste Sprache, um semantische Informationen über beliebige Ressourcen zu repräsentieren.
- Features:
  - ⊗ - Klassen, Eigenschaften, Range- und Domaineinschränkungen
  - - Vererbungshierarchien für Klassen und Eigenschaften
- Fehlende Unterstützung von:
  - - Datentypen
  - - Aufzählungen
  - - Stärkere Einschränkung von Eigenschaften
- Als Basis für die Entwicklung von OWL wurden die schon bestehenden Ontologiesprachen DAML-ONT (Darpa Agent Markup Language - Ontology) und OIL (Ontology Inference Layer) benutzt und zu DAML + OIL verschmolzen.
- Technisch basiert OWL somit auf RDF und historisch auf DAML+OIL und wurde so zur Standard-Ontologiesprache für das Semantic Web (10. Februar 2004).

# Web Ontology Language (OWL)

## Ausprägungen von OWL

Die OWL Sprache stellt drei Untersprachen mit zunehmender Ausdruckstärke bereit. Diese Unterteilung wurde gemacht, um den Bedürfnissen unterschiedlicher Gruppen von Benutzern entgegen zu kommen.

- OWL Lite
- OWL DL (Description Logics)
- OWL Full

# Web Ontology Language (OWL)

## OWL Lite

OWL Lite ist der einfachste Vertreter der drei Untersprachen und richtet sich vorwiegend an Benutzer die eine einfache Klassifikationshierarchie und einfache Restriktionsmöglichkeiten benötigen. OWL Lite ist eine Erweiterung einer eingeschränkten Sicht auf RDF.

Features:

- Beinhaltet alle grundlegenden Funktionen von OWL
- Relativ einfache Programmierung von Tools
- Kann von Benutzern einfach verstanden werden.

# Web Ontology Language (OWL)

## OWL DL (Description Logic)

OWL DL ist eine eingeschränkte Version von OWL Full. Sie bietet maximale Ausdrucksstärke bei gleichzeitiger Zusicherung, dass Daten in endlicher Zeit ausgewertet werden können. OWL DL ist eine Erweiterung einer eingeschränkten Sicht auf RDF.

Features:

- maximale Ausdrucksstärke / komplexe Restriktionen
- Zusicherung, dass Inferenzmaschinen Daten in endlicher Zeit auswerten können
- Erlaubt alle OWL Sprachkonstrukte, diese können jedoch nur unter bestimmten Bedingungen verwendet werden.
- Genügt den Anforderungen der Description Logic

# Web Ontology Language (OWL)

## OWL Full

OWL Full ist die ausdrucksstärkste Version der OWL Sprachen. Sie kann als eine Erweiterung von RDF gesehen werden.

Features:

- maximale Ausdrucksstärke
- Die Auswertbarkeit ist nicht garantiert
- Erlaubt die Bedeutung von vordefiniertem (RDF und OWL) Vokabular zu erweitern
- Unwahrscheinlich, dass eine Schlussfolgerungssoftware jedes Feature von OWL Full unterstützen wird

# Web Ontology Language (OWL)

## OWL Beziehungen

Jede der drei OWL Untersprachen ist eine Erweiterung ihres Vorgängers. Auf grund dieser Hierarchie treffen folgende Aussagen zu. Ihre Umkehrungen jedoch nicht.

- Jede legale OWL Lite Ontologie ist eine legale OWL DL Ontologie
- Jede legale OWL DL Ontologie ist eine legale OWL Full Ontologie
  
- Jede gültige OWL Lite Folgerung ist eine gültige OWL DL Folgerung
- Jede gültige OWL DL Folgerung ist eine gültige OWL Full Folgerung

# Web Ontology Language (OWL)

## Aufbau eines OWL Dokuments

Ein OWL Dokument setzt sich aus folgenden Bestandteilen zusammen:

- Namespace Deklarationen
- Ontologie Header
- Ontologie

# Web Ontology Language (OWL)

## Aufbau eines OWL Dokuments

Die Namespace Deklaration legt fest, welche Vokabularien im OWL Dokument benutzt wird.

```
<rdf:RDF
  xmlns      = "http://www.ckoehn.de/seminar#"
  xml:base   = "http://www.ckoehn.de/seminar#"
  xmlns:owl  = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf  = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd  = "http://www.w3.org/2001/XMLSchema#">

  <Ontologie Header>

  <Ontologie>
</rdf:RDF>
```

# Web Ontology Language (OWL)

## Aufbau eines OWL Dokuments

Der Ontologie Header bindet Metainformationen wie Kommentierung, Versionskontrolle und den Import von anderen Ontologien ein.

```
<rdf:RDF
  xmlns      = "http://www.ckoehn.de/seminar#"
  xml:base   = "http://www.ckoehn.de/seminar#"
  xmlns:owl  = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf  = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd  = "http://www.w3.org/2001/XMLSchema#">

  <owl:Ontology rdf:about="">
    <rdfs:comment>Meine Seminar-Ontologie</rdfs:comment>
    <owl:priorVersion rdf:resource="http://www.ckoehn.de/seminar.01#" />
    <owl:imports rdf:resource="http://www.ckoehn.de/eineAndere#" />
    <rdfs:label>Kuenstler Ontologie</rdfs:label>
  </owl:Ontology>

  <Ontologie>
</rdf:RDF>
```

# Web Ontology Language (OWL)

## Sprachkonstrukte

Die Grundbausteine der OWL Syntax lassen sich in vier Bereiche einteilen:

- Klassen und Individuen
- Eigenschaften
- Merkmale von Eigenschaften
- Einschränkungen von Eigenschaften

Anhand eines Beispiels werden die wichtigsten Bausteine von OWL vorgestellt.

# Web Ontology Language (OWL)

## Klassen

Die Basis einer jeden Ontologie bilden sogenannte *Benannte Klassen*. Jede erstellte Klasse ist implizit eine Unterklasse von *owl:Thing*. *owl:Nothing* stellt eine definierte leere Klasse dar.

```
<owl:Class rdf:ID="Kunstwerk" />
<owl:Class rdf:ID="Kuenstler" />
<owl:Class rdf:ID="Gebaeude" />
<owl:Class rdf:ID="Technik" />
```

Diese Klassen bilden unsere Basis für die Definition der Unterklassen *Bild*, *Skulptur*, *Bildhauer*, *Maler*, *Museum* und *Galerie*.

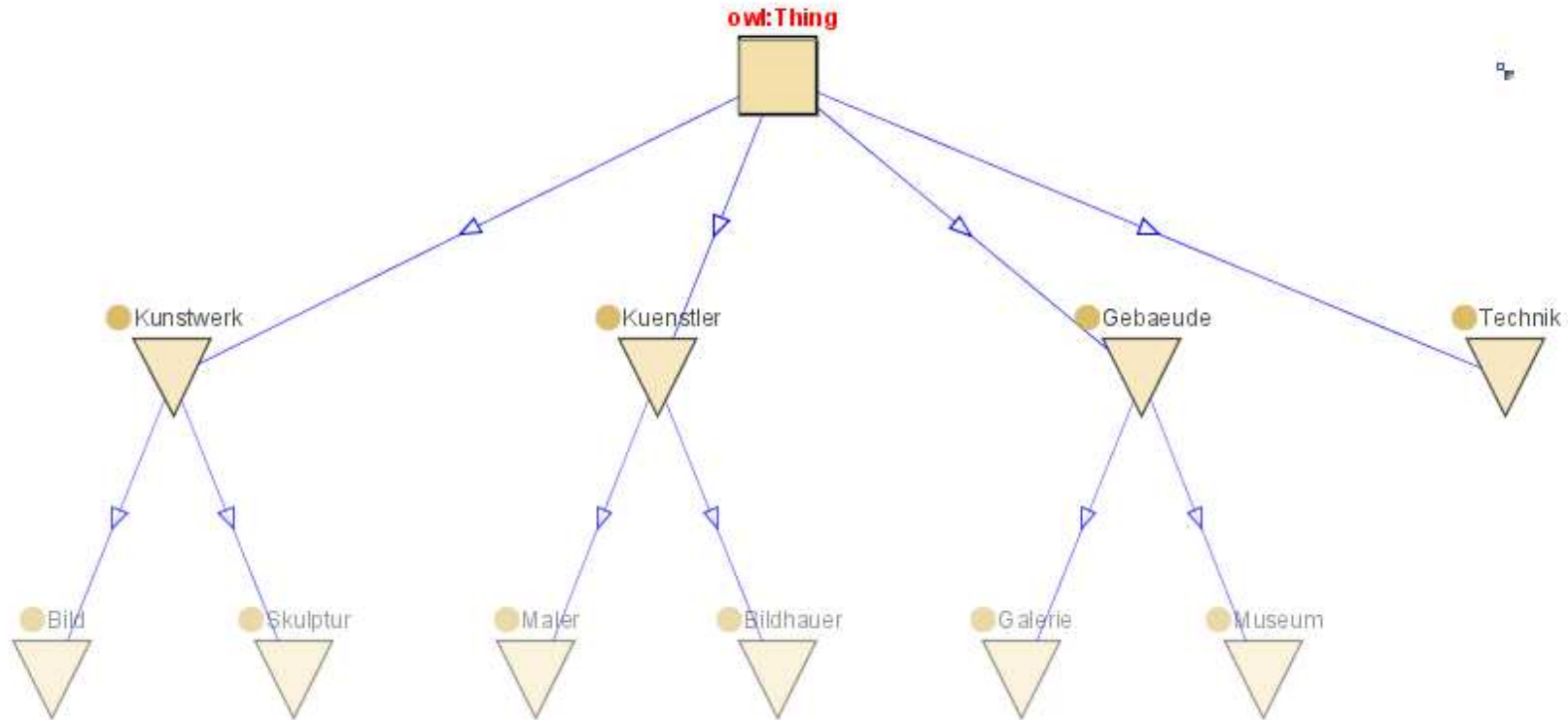
```
<owl:Class rdf:ID="Bild">
  <rdfs:subClassOf rdf:resource="#Kunstwerk" />
</owl:Class>

...
```

# Web Ontology Language (OWL)

## Übersicht

Die von uns definierte Ontologie sieht nun folgendermaßen aus:



# Web Ontology Language (OWL)

## Individuen/Instanzen

Individuen bzw Mitglieder der eben definierten Klassen, lassen sich wie folgt beschreiben.

```
<Maler rdf:ID="Rembrandt" />
<Maler rdf:ID="Friedrich" />

<Bild rdf:ID="FrauImFenster" />
<Bild rdf:ID="DerSommer" />
<Bild rdf:ID="DieNachtwache" />

<owl:Thing rdf:ID="Rembrandt">
  <rdf:type rdf:resource="#Maler" />
</owl:Thing>
```

In OWL sind Individuen mit unterschiedlichen Namen, per se nicht explizit unterschiedlich.

# Web Ontology Language (OWL)

## Eigenschaften

In OWL wird zwischen zwei verschiedene Arten von Eigenschaften unterschieden.

- Datatype properties
  - - Verbinden Individuen mit XML-Schema Datentypen und RDF-Literalen
- Object properties
  - - Verbinden Individuen mit anderen Individuen

# Web Ontology Language (OWL)

## Eigenschaften

*Object properties* können Domain-, sowie Rangeangaben enthalten. Diese sagen aus, dass Instanzen der Klasse *Kuenstler* mit Instanzen der Klasse *Kunstwerk* in Verbindung gebracht werden.

```
<owl:ObjectProperty rdf:ID="erzeugt">
  <rdfs:range rdf:resource="#Kunstwerk"/>
  <rdfs:domain rdf:resource="#Kuenstler"/>
</owl:ObjectProperty>
```

*Datatype properties* können ebenso Domain- und Rangeangaben enthalten. Die Rangeinformation besteht hierbei jedoch aus XML Schema Datentypen oder aus einer *rdfs:Literal* Angabe.

```
<owl:DatatypeProperty rdf:ID="name">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

# Web Ontology Language (OWL)

## Eigenschaften

Auch von Eigenschaften lassen sich Untereigenschaften bilden. Wir schränken hier die Domain- und die Rangeangabe von *erzeugt* für Maler und Bildhauer noch weiter ein.

```
<owl:ObjectProperty rdf:ID="malt">
  <rdfs:subPropertyOf rdf:resource="#erzeugt"/>
  <rdfs:domain rdf:resource="#Maler"/>
  <rdfs:range rdf:resource="#Bild"/>
</owl:ObjectProperty>

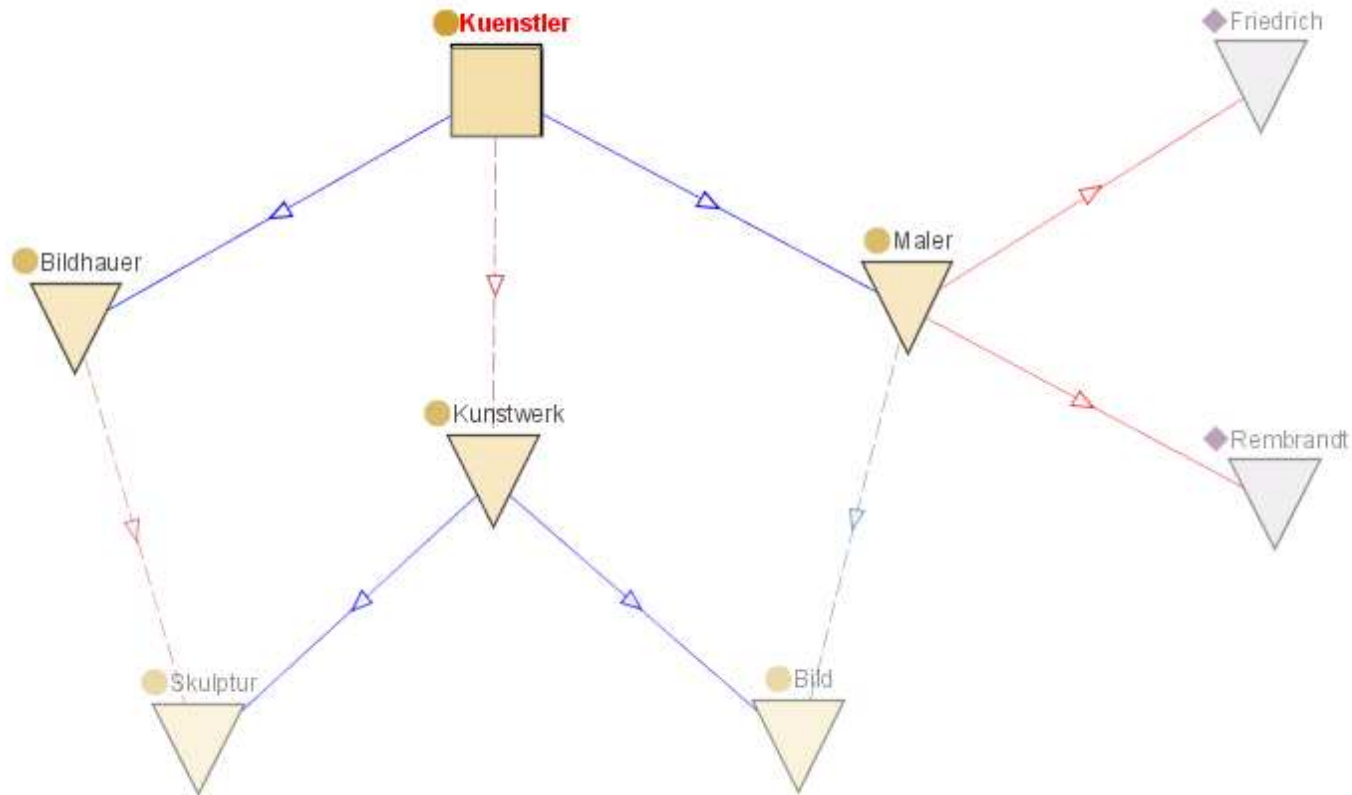
<owl:ObjectProperty rdf:ID="schlaegt">
  <rdfs:subPropertyOf rdf:resource="#erzeugt"/>
  <rdfs:domain rdf:resource="#Bildhauer"/>
  <rdfs:range rdf:resource="#Skulptur"/>
</owl:ObjectProperty>
```

```
<Maler rdf:ID="Rembrandt">
  <malt rdf:resource="#DieNachtwache" />
  <name rdf:datatype="&xsd:string">Rembrandt Harmenszoon van Rijn</name>
</Maler>
...
```

# Web Ontology Language (OWL)

## Übersicht

Der von uns bearbeitete Teil sieht nun wie folgt aus:



# Web Ontology Language (OWL)

## Eigenschaften - Merkmale

Um Eigenschaften noch genauer spezifizieren zu können, stellt OWL sogenannte *Property Characteristica* bereit. Diese erlauben ein erweitertes Folgern aus Eigenschaften.

Merkmale:

- TransitiveProperty
  - - *hatVorfahre* ist transitiv.
- SymmetricProperty
  - - *hatNachbar* oder *hatGeschwister* sind symmetrisch.
- FunctionalProperty
  - - *hatAlter* oder *hatMutter* sind funktional.
- inverseOf
  - - *erzeugtVon*, *gemaltVon* oder *geschlagenVon* sind Inverse zu den eben definierten Eigenschaften.
- InverseFunctionalProperty
  - - *hatMutter* und *istMutterVon* sind invers funktional zueinander.

# Web Ontology Language (OWL)

## Eigenschaften - Merkmale

Mit *owl:inverseOf* lässt sich ausdrücken, dass eine Eigenschaft das Gegenteil einer anderen Eigenschaft ist. In unserem Beispiel wollen wir dies nutzen um die Eigenschaften *erzeugtVon*, *geschlagenVon* und *gemaltVon* zu definieren.

```
<owl:ObjectProperty rdf:ID="erzeugtVon">
  <owl:inverseOf rdf:resource="#erzeugt"/>
  <rdfs:domain rdf:resource="#Kunstwerk"/>
  <rdfs:range rdf:resource="#Kuenstler"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="geschlagenVon">
  <rdfs:subPropertyOf rdf:resource="#erzeugtVon"/>
  <owl:inverseOf rdf:resource="#schlaegt"/>
  <rdfs:domain rdf:resource="#Skulptur"/>
  <rdfs:range rdf:resource="#Bildhauer"/>
</owl:ObjectProperty>

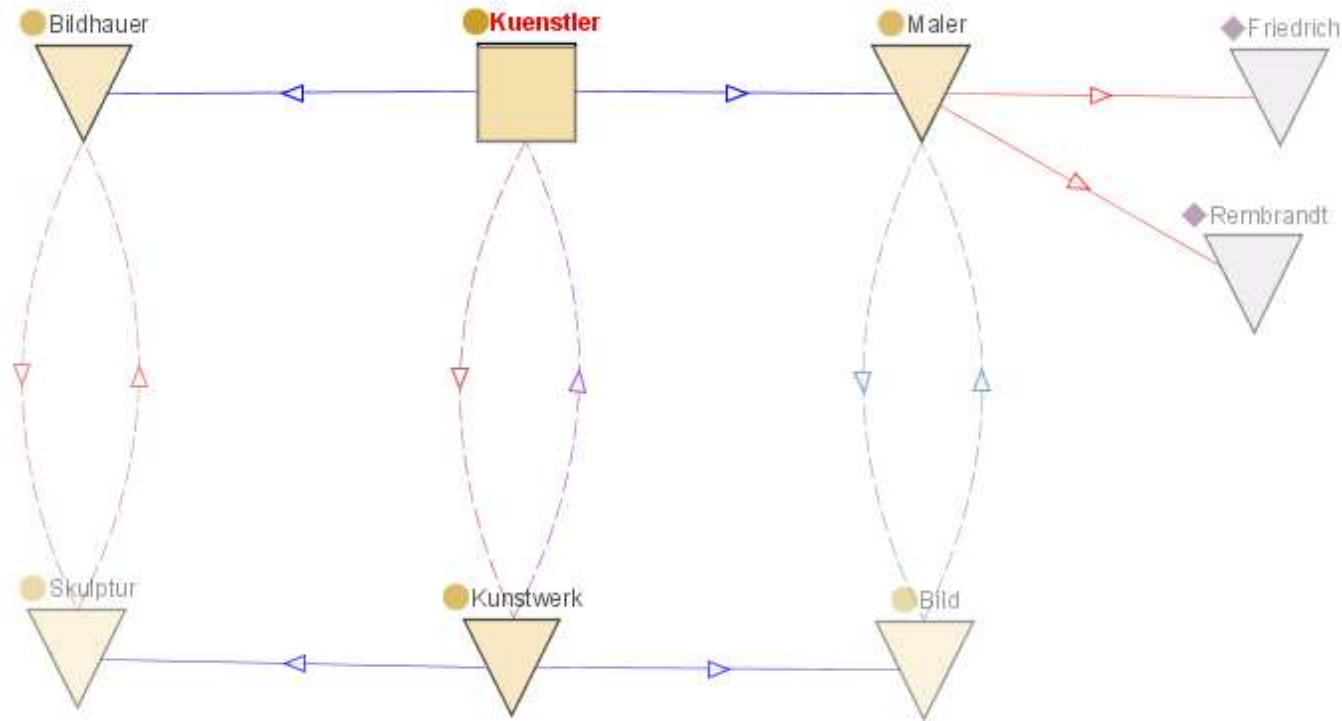
...
```

Im Gegensatz zur *normalen* Eigenschaft, sind die Domain- und Rangeangaben bei der Inversen vertauscht.

# Web Ontology Language (OWL)

## Übersicht

Der von uns bearbeitete Teil sieht nun wie folgt aus:



# Web Ontology Language (OWL)

## Eigenschaften - Einschränkungen

*Property Restrictions* sind in OWL vorgesehen, um den Wertebereich einer Eigenschaft in spezifischen Kontexten einschränken zu können.

Einschränkungen:

- `allValuesFrom`
- `someValuesFrom`
- `cardinality`
- `hasValue`

# Web Ontology Language (OWL)

## Eigenschaften - Einschränkungen

*owl:allValuesFrom* sagt aus, dass für die angegebene Eigenschaft, nur Werte aus einem bestimmten Bereich zulässig sind. Hierbei müssen *alle* Werte Mitglieder dieser Klasse sein.

Wir wollen unser Beispiel dahingehend ändern, dass *erzeugt* keine globale Restriktion mehr besitzt.

```
<owl:Class rdf:ID="Kuenstler">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#erzeugt"/>
      <owl:allValuesFrom rdf:resource="#Kunstwerk"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

*owl:someValuesFrom* fordert, dass *mindestens ein* Wert Mitglied der angegebenen Klasse sein muss.

# Web Ontology Language (OWL)

## Eigenschaften - Einschränkungen

Mit Hilfe von *owl:cardinality* lässt sich die exakte Anzahl einer Eigenschaft in Bezug auf die Mitglieder der Klasse beschränken. Mit *owl:minCardinality* und *owl:maxCardinality* lässt sich eine ober und eine untere Grenze definieren.

```
<owl:Class rdf:ID="Kuenstler">
...
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#erzeugt"/>
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
  </owl:Restriction>
</rdfs:subClassOf>
...
</owl:Class>
```

# Web Ontology Language (OWL)

## Eigenschaften - Einschränkungen

*owl:hasValue* kann benutzt werden, um Klassen zu konstruieren die auf der Existenz eines Eigenschaftswerts basieren.

```
<owl:Class rdf:ID="RembrandtsBilder">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#erzeugtVon" />
      <owl:hasValue rdf:resource="#Rembrandt" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Wir erzeugen hier eine Klasse, deren Mitglieder alle von Rembrandt gemalten Bilder sind.

# Web Ontology Language (OWL)

## Äquivalenzen

Bei der Entwicklung einer neuen Ontologie sollte sich ein Entwickler umschauen und so evtl schon vorhandene Ontologien nutzen, um seine zu modellieren. Hierbei gibt es oft die Notwendigkeit zu sagen, dass zwei Individuen, Klassen oder Eigenschaften das Gleiche beschreiben.

Klassen und Eigenschaften:

- *owl:equivalentClass*
- *owl:equivalentProperty*

Instanzen/Individuen:

- *owl:sameAs*
- *owl:differentFrom*
- *owl:AllDifferentFrom*

# Web Ontology Language (OWL)

## Komplexe Klassen

In OWL ist es möglich durch Benutzung der Mengenoperationen oder Aufzählungen neue Klassen zu konstruieren. Hierbei ist noch besonders hervorzuheben, dass Klassen explizit als *verschieden* deklariert werden können, damit Individuen nicht gleichzeitig Mitglied dieser beiden Klassen sein können.

Möglich wird dies durch:

- *owl:intersectionOf*
- *owl:unionOf*
- *owl:complementOf*
  
- *owl:oneOf*
  - Legt die Mitglieder der Klasse explizit fest. Keine anderen mehr möglich
  
- *owl:disjointWith*
  - Alle angegebenen Klassen sind voneinander verschieden

# Web Ontology Language (OWL)

## Tools

### Reasoner

- FaCT++
- Racer
- ...

### Editoren

- Protégé OWL
- Eclipse Plugin SWeDE
- ...

# Web Ontology Language (OWL)

## Reasoning - Beispiel

```
Individual(a:Walt type(a:person)
value(a:has_pet a:Hui)
  value(a:has_pet a:Lui)
  value(a:has_pet a:Dui))

Individual(a:Hui type(a:duck))
Individual(a:Dui type(a:duck))
Individual(a:Lui type(a:duck))

DifferentIndividuals(a:Hui a:Dui a:Lui)

Class(a:animal_lover complete intersectionOf(a:person restriction(a:has_pet minCardinality(3))))
```

Folgerungen hieraus sind:

- Walt hat die Haustiere Hui, Lui und Dui
- Hui, Lui und Dui sind alle unterschiedliche Individuen
- Walt hat drei Haustiere und ist deshalb ein Tierfreund

**Vielen Dank für die Aufmerksamkeit**