

# XML-Datenmodellierung und Web-Services

---

- XML
  - Datenmodell
  - Schemabeschreibungssprachen
  - Anfragesprachen: XPath und XQuery
- Web-Services
  - Überblick
  - WSDL
  - UDDI
  - SOAP

# **XML: Extensible Markup Language**

---

**Datenmodell**

**Schemabeschreibung**

**Anfragesprachen**

# HTML-Datenmodell

```
<UL>
```

```
<LI> Curie
```

```
<LI> Sokrates
```

```
</UL>
```

```
<UL>
```

```
<LI> Mäeutik
```

```
<LI> Bioethik
```

```
</UL>
```

- Kein Schema
- Nur Insider können die beiden Listen interpretieren
  - Oben: Professoren
  - Unten: Vorlesungen
- Wenig geeignet als Datenaustauschformat
  - Man muß irgendwie dann auch mitschicken, was damit gemeint ist
- Verarbeitung von HTML-Daten
  - „Screen-scraping“
  - Wrapper

# Relationales Datenmodell

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Vorlesungen			
VorlNr	Titel	SWS	Gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

- Schema ist vorgegeben und man kann nur schema-konforme Daten einfügen (Problem Ausnahmen → null-Werte)
- Bedeutung der Daten wird durch das Schema definiert
- Kein Datenaustauschformat

# XML-Datenmodell

- Liegt irgendwo dazwischen
  - HTML
    - Schema-los
    - Beliebige Daten, solange Syntax stimmt
  - Relationen
    - Schema
    - Keine Abweichungen
- Semi-strukturierte Daten
  - Teilweise schematisch
  - Aber Ausnahmen
  - Wenn Schema, dann muss es eingehalten werden

# Unsere Beispiel-Daten in XML ...

**<Professoren>**

**<ProfessorIn>Curie</ProfessorIn>**

**<ProfessorIn>Sokrates</ProfessorIn>**

**</Professoren>**

**Semantische Tags (Marken):  
Geben die Bedeutung der  
Elemente an, immer paarweise**

**<...> Element </...>**

**<Vorlesungen>**

**<Vorlesung>Mäeutik</Vorlesung>**

**<Vorlesung>Bioethik</Vorlesung>**

**</Vorlesungen>**

# XML-Daten mit Schema (DTD)

```
<?xml version="1.0" encoding='ISO-8859-1'?>
```

```
<!-- obige Zeile ist der Prolog, diese Zeile ist ein Kommentar -->
```

```
<!-- Schema als DTD -->
```

```
<!DOCTYPE Buch[
```

```
  <!ELEMENT Buch (Titel, Autor*, Verlag)>
```

```
  <!ATTLIST Buch Jahr CDATA #REQUIRED>
```

```
  <!ELEMENT Titel (#PCDATA)>
```

```
  <!ELEMENT Autor (#PCDATA)>
```

```
  <!ELEMENT Verlag (#PCDATA)>
```

```
<!-- Wurzelement-->
```

```
<Buch Jahr="2001">
```

```
  <Titel>Datenbanksysteme: Eine Einführung</Titel>
```

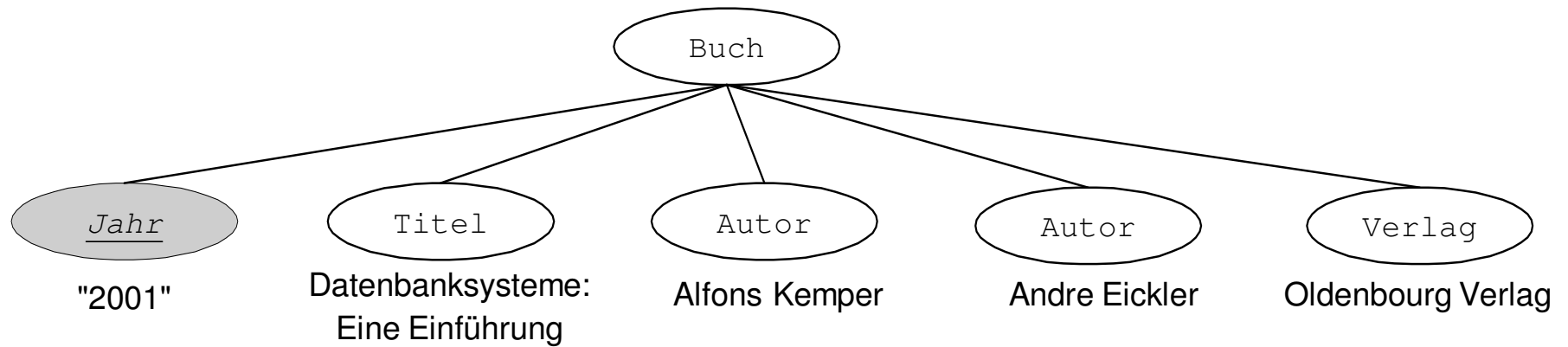
```
  <Autor>Alfons Kemper</Autor>
```

```
  <Autor>Andre Eickler</Autor>
```

```
  <Verlag>Oldenbourg Verlag</Verlag>
```

```
</Buch>
```

# Die hierarchische Struktur im Bild





# Rekursive Strukturen

```
<?xml version="1.0" encoding='ISO-8859-1'?>
```

```
<!-- Schema als DTD -->
```

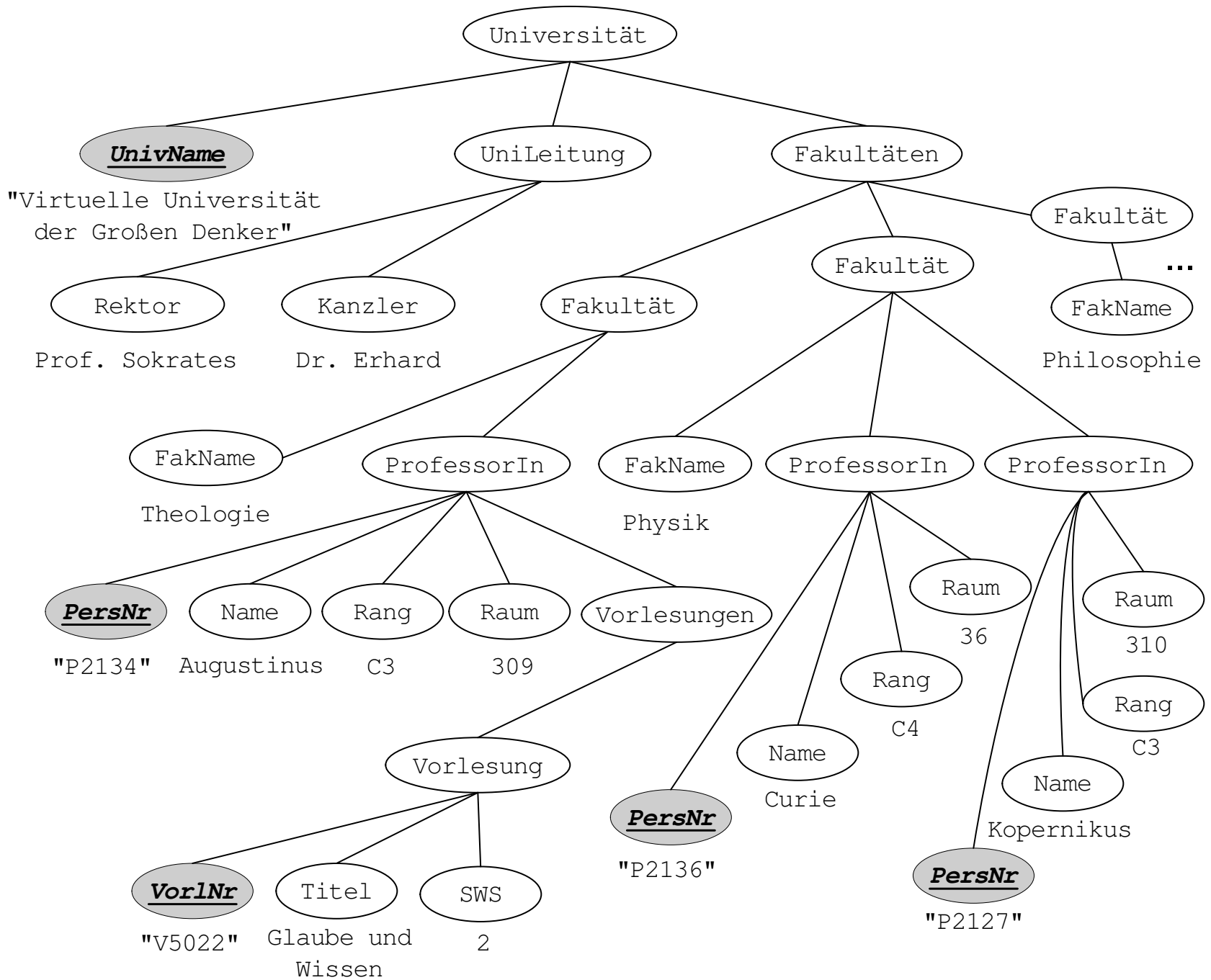
```
<!DOCTYPE Bauteil[  
  <!ELEMENT Bauteil (Beschreibung, Bauteil*)>  
  <!ATTLIST Bauteil Preis CDATA #REQUIRED>  
  <!ELEMENT Beschreibung (#PCDATA)>  

```

```
<!-- Wurzelement-->
```

```
<Bauteil Preis="350000">  
  <Beschreibung>Maybach 620 Limousine</Beschreibung>  
  <Bauteil Preis="50000">  
    <Beschreibung>V12-Biturbo Motor mit 620 PS</Beschreibung>  
    <Bauteil Preis="2000">  

```



```
<?xml version="1.0" encoding='ISO-8859-1'?>
<Universität UnivName="Virtuelle Universität der Großen Denker">
<UniLeitung>
  <Rektor>Prof. Sokrates</Rektor>
  <Kanzler>Dr. Erhard</Kanzler>
</UniLeitung>
<Fakultäten>
  <Fakultät>
    <FakName>Theologie</FakName>
    <ProfessorIn PersNr="2134">
      <Name>Augustinus</Name>
      <Rang>C3</Rang>
      <Raum>309</Raum>
      <Vorlesungen>
        <Vorlesung VorlNr="5022">
          <Titel>Glaube und Wissen</Titel>
          <SWS>2</SWS>
        </Vorlesung>
      </Vorlesungen>
    </ProfessorIn>
  </Fakultät>
```

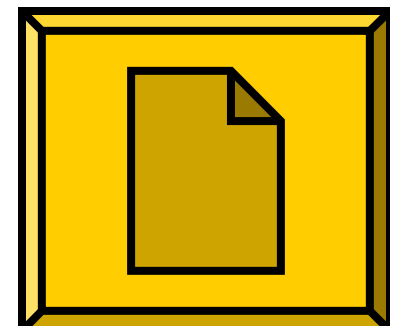
# XML- Dokument der Universität



```
<Fakultät>
  <FakName>Physik</FakName>
  <ProfessorIn PersNr="2136">
    <Name>Curie</Name>
    <Rang>C4</Rang>
    <Raum>36</Raum>
  </ProfessorIn>
  <ProfessorIn PersNr="2127">
    <Name>Kopernikus</Name>
    <Rang>C3</Rang>
    <Raum>310</Raum>
  </ProfessorIn>
</Fakultät>

<Fakultät>
  <FakName>Philosophie</FakName>
  ...
  ...
</Fakultät>
</Fakultäten>
</Universität>
```

Im Browser



# XML Namensräume

...

```
<Universität xmlns="http://www.db.uni-passau.de/Universitaet"  
  UnivName="Virtuelle Universität der Großen Denker">  
  <UniLeitung>
```

...

# XML Namensräume

```
...  
<Universität xmlns="http://www.db.uni-passau.de/Universitaet"  
  xmlns:lit="http://www.db.uni-passau.de/Literatur"  
  UnivName="Virtuelle Universität der Großen Denker">  
  <UniLeitung>
```

```
...  
  <Vorlesung>  
    <Titel> Informationssysteme </Titel>  
    ...  
    <lit:Buch lit:Jahr="2004">  
      <lit:Titel>Datenbanksysteme: Eine Einführung</lit:Titel>  
      <lit:Autor>Alfons Kemper</lit:Autor>  
      <lit:Autor>Andre Eickler</lit:Autor>  
      <lit:Verlag>Oldenbourg Verlag</lit:Verlag>  
    </lit:Buch>  
  </Vorlesung>
```

```
...
```

# XML-Schema: mächtiger als DTDs

```
<?xml version="1.0" encoding='ISO-8859-1'?>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  targetNamespace="http://www.db.uni-passau.de/Universitaet">
```

```
<xsd:element name="Universität" type="UniInfoTyp"/>
```

```
<xsd:complexType name="UniInfoTyp">
```

```
<xsd:sequence>
```

```
<xsd:element name="UniLeitung">
```

```
<xsd:complexType>
```

```
<xsd:sequence>
```

```
<xsd:element name="Rektor" type="xsd:string"/>
```

```
<xsd:element name="Kanzler" type="xsd:string"/>
```

```
</xsd:sequence>
```

```
</xsd:complexType>
```

```
</xsd:element>
```

```
<xsd:element name="Fakultäten">
```



# XML-Schema: mächtiger als DTDs

```
<xsd:element name="Fakultäten">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="Fakultät"  
        minOccurs="0"  
        maxOccurs="unbounded"  
        type="Fakultätentyp"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>  
</xsd:sequence>  
  <xsd:attribute name="UnivName" type="xsd:string"/>  
</xsd:complexType>
```





```
<xsd:complexType name="FakultätenTyp">  <xsd:sequence>
  <xsd:element name="FakName" type="xsd:string"/>
  <xsd:element name="ProfessorIn" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="Rang" type="xsd:string"/>
        <xsd:element name="Raum" type="xsd:integer"/>
        <xsd:element name="Vorlesungen" minOccurs="0" type="VorInfo"/>
      </xsd:sequence>
      <xsd:attribute name="PersNr" type="xsd:ID"/>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence> </xsd:complexType>
```



```
<xsd:complexType name="VorlInfo">  <xsd:sequence>
  <xsd:element name="Vorlesung" minOccurs="1" maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Titel" type="xsd:string"/>
        <xsd:element name="SWS" type="xsd:integer"/>
      </xsd:sequence>
      <xsd:attribute name="VorlNr" type="xsd:ID"/>
      <xsd:attribute name="Voraussetzungen" type="xsd:IDREFS"/>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence> </xsd:complexType>

</xsd:schema>
```

# Verweise in XML-Dokumenten

- XML ist „super“ für die Modellierung von Hierarchien
- Entsprechen den geschachtelten Elementen
- Genau das hatten wir in dem Uni-Beispiel

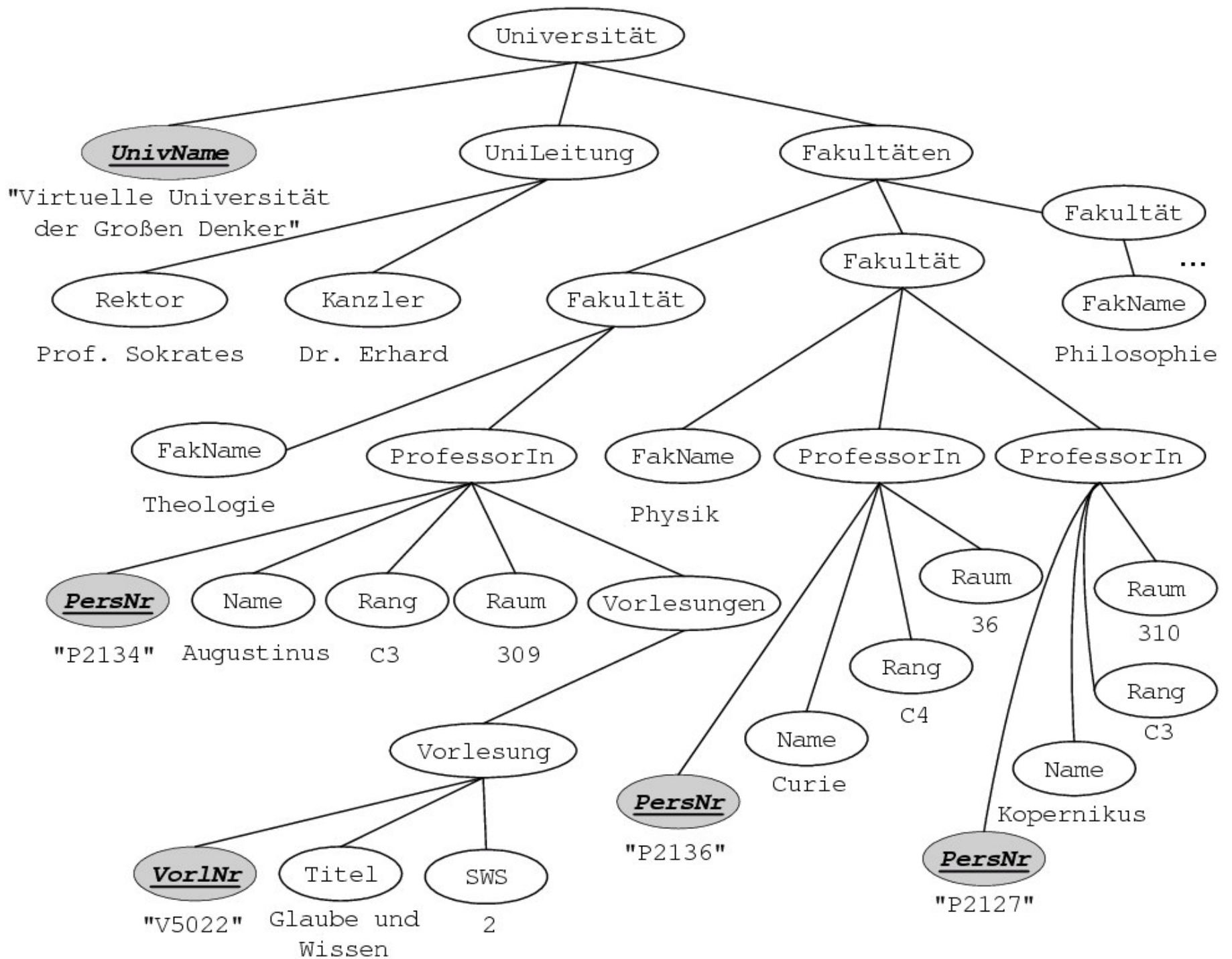


**Uni**

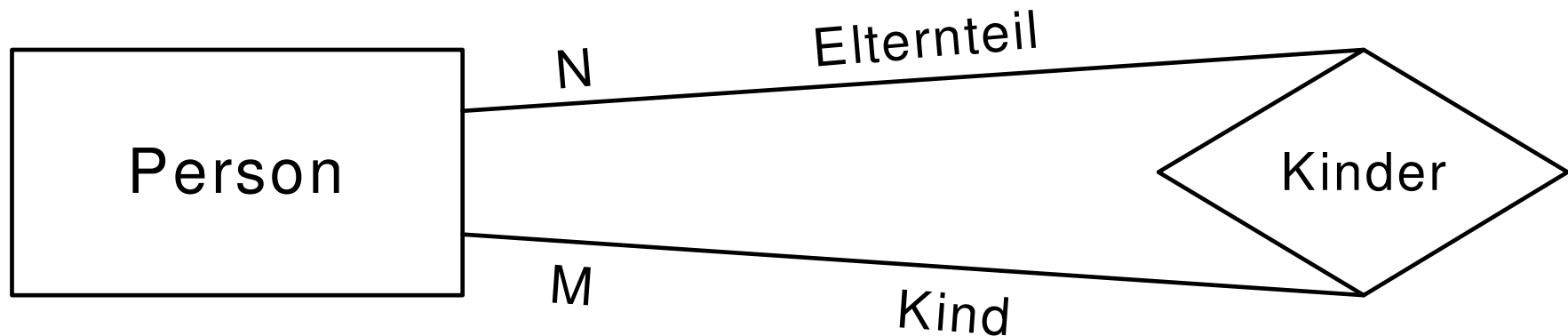
**Fakultäten**

**Professoren**

**Vorlesungen**

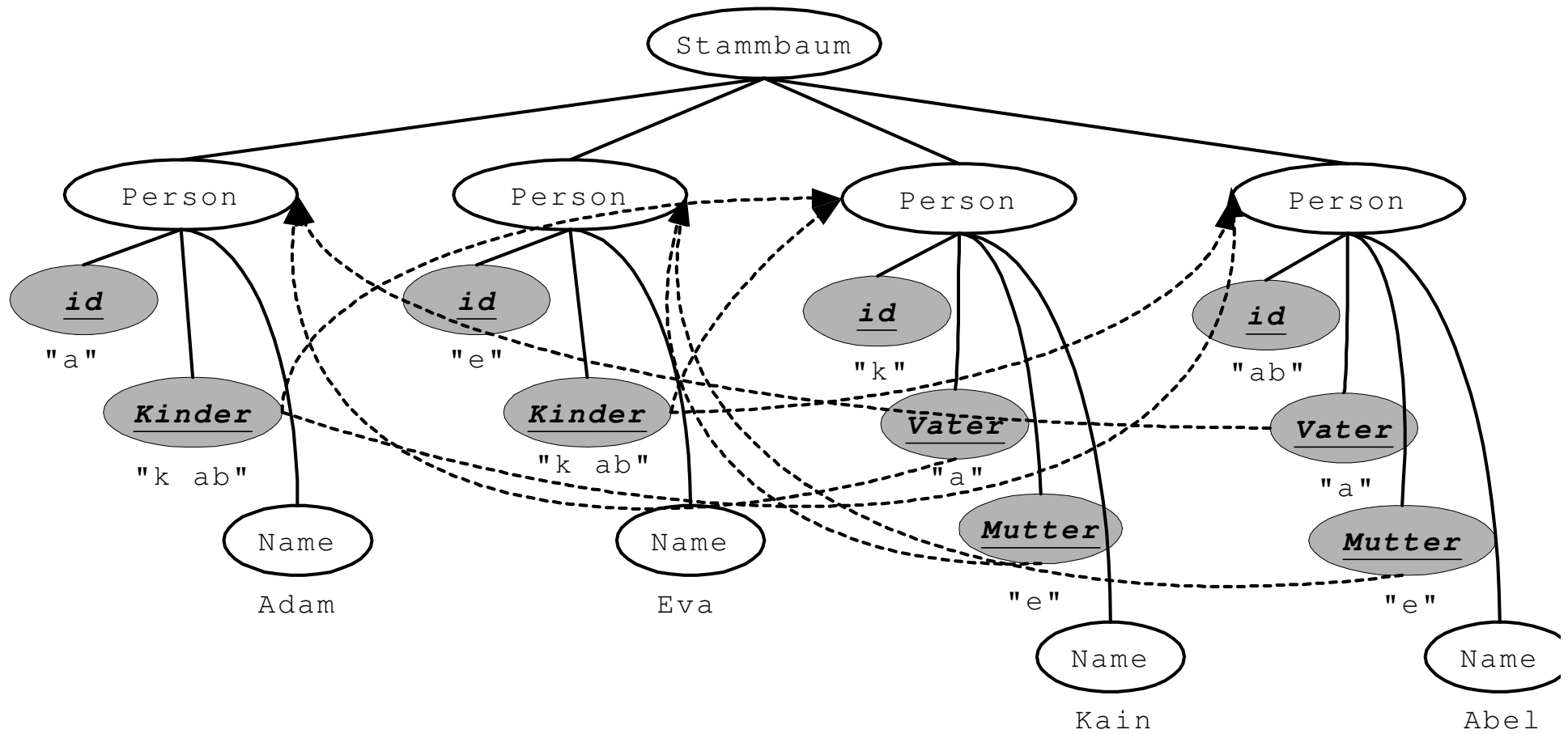


# Modellierung des Stammbaums



- Kinder haben zwei Elternteile
  - Also kann man ein Kind nicht mehr als Sub-Element einer Person modellieren
  - Wem soll man es zuordnen (Vater oder Mutter)
  - Also muss man mit Verweisen (IDREF und IDREFS) „arbeiten“

# Graphische Darstellung des XML-Dokuments



# Familie.xml

```
<!DOCTYPE Stammbaum [  
  <!ELEMENT Stammbaum (Person*)>  
  <!ELEMENT Person (Name)>  
  <!ELEMENT Name (#PCDATA)>  
    <!ATTLIST Person id      ID      #REQUIRED  
      Mutter IDREF #IMPLIED  
      Vater  IDREF #IMPLIED  
      Kinder IDREFS #IMPLIED> ]>
```

```
<Stammbaum>  
  <Person id="a" Kinder="k ab">  
    <Name>Adam</Name> </Person>  
  <Person id="e" Kinder="k ab">  
    <Name>Eva</Name> </Person>  
  <Person id="k" Mutter="e" Vater="a">  
    <Name>Kain</Name> </Person>  
  <Person id="ab" Mutter="e" Vater="a">  
    <Name>Abel</Name> </Person>  
</Stammbaum>
```

# XML-Anfragesprache XQuery

- Basiert auf Xpath, einer Sprache für Pfadausdrücke
- Ein Lokalisierungspfad besteht aus einzelnen Lokalisierungsschritten
- Jeder Lokalisierungsschritt besteht aus bis zu drei Teilen:
  - Achse::Knotentest[Prädikat]

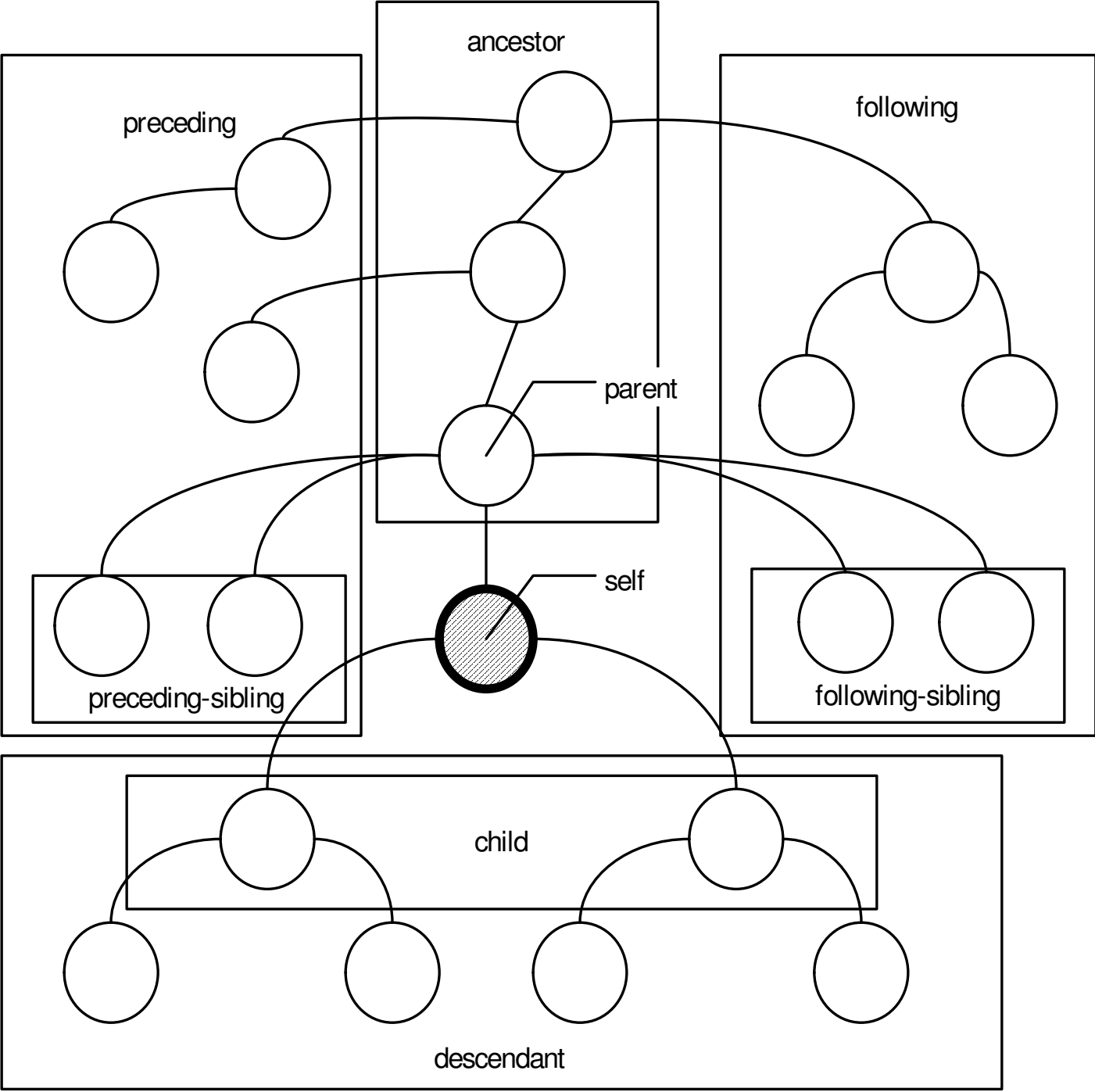


# XPath-Achsen

- self: Hierbei handelt es sich um den Referenzknoten.
- attribute: Hierunter fallen alle Attribute des Referenzknotens -- falls er überhaupt welche besitzt.
- child: Entlang dieser Achse werden alle direkten Unterelemente bestimmt.
- descendant: Hierunter fallen alle direkten und indirekten Unterelemente, also die Kinder und deren Kinder u.s.w.
- descendant-or-self: Wie oben, außer dass der Referenzknoten hier auch dazu gehört.
- parent: Der Vaterknoten des Referenzknotens wird über diese Achse ermittelt.
- ancestor: Hierzu zählen alle Knoten auf dem Pfad vom Referenzknoten zur Wurzel des XML-Baums.

# Achsen ... cont'd

- ancestor-or-self: Wie oben, außer dass der Referenzknoten auch mit eingeschlossen wird.
- following-sibling: Dies sind die in Dokumentreihenfolge nachfolgenden Kinder des Elternknotens von self.
- preceding-sibling: Hierbei handelt es sich um die in Dokumentreihenfolge vorangehenden Kinder des Elternknotens von self.
- following: Alle Knoten, die in Dokumentreihenfolge nach dem Referenzknoten aufgeführt sind. Die Nachkommen (descendant) des Referenzknotens gehören aber nicht dazu.
- preceding: Alle Knoten, die im Dokument vor dem Referenzknoten vorkommen -- allerdings ohne die Vorfahren (ancestor).



# XPath: Beispiele

```
doc("Uni.xml")/child::Universität[self::*/*attribute::UnivName=  
    "Virtuelle Universität der Großen Denker"]
```

- `<FakName>Theologie</FakName>`
- `<FakName>Physik</FakName>`
- `<FakName>Philosophie</FakName>`
  
- Äquivalent für unser Beispiel:
  - `doc("Uni.xml")/descendant-or-self::FakName`

# XPath: Beispiele

- `doc("Uni.xml")/child::Universität/attribute::UnivName`
- `UnivName="Virtuelle Universität der Großen Denker"`

```
doc("Uni.xml")/child::Universität/child::Fakultäten/  
  child::Fakultät[self::*/*child::FakName="Theologie"]/  
    descendant-or-self::Vorlesung/child::Titel
```

# XPath: Beispiele

```
doc("Uni.xml")/child::Universität/child::Fakultäten/  
  child::Fakultät[child::FakName="Philosophie"]/  
    child::ProfessorIn[child::Rang="C4"]/child::Vorlesungen/  
      child::Vorlesung/child::Titel
```

```
<Titel>Ethik</Titel> <Titel>Mäeutik</Titel> <Titel>Logik</Titel>  
<Titel>Erkenntnistheorie</Titel> <Titel>Wissenschaftstheorie</Titel>  
<Titel>Bioethik</Titel> <Titel>Grundzüge</Titel> <Titel>Die 3  
  Kritiken</Titel>
```

# XPath: Beispiel

```
doc("Uni.xml")/child::Universität/child::Fakultäten/  
    child::Fakultät/child::FakName[parent::Fakultät/  
        child::ProfessorIn/child::Vorlesungen]
```

```
<FakName>Theologie</FakName>
```

```
<FakName>Philosophie</FakName>
```

# XPath: Beispiele

```
doc("Uni.xml")/child::Universität/child::Fakultäten/  
child::Fakultät[position()=2]
```

wird also die zweite Fakultät ausgegeben:

```
<Fakultät>  
  <FakName>Physik</FakName>  
  <ProfessorIn PersNr="P2136">  
    <Name>Curie</Name>  
    <Rang>C4</Rang>  
    <Raum>36</Raum>  
  </ProfessorIn>  
  <ProfessorIn PersNr="P2127">  
    <Name>Kopernikus</Name>  
    <Rang>C3</Rang>  
    <Raum>310</Raum>  
  </ProfessorIn>  
</Fakultät>
```



# XPath: Beispiele

```
doc("Uni.xml")/child::Universität/child::Fakultäten/  
  child::Fakultät[child::ProfessorIn/child::Vorlesungen/  
    child::Vorlesung/child::Titel="Mäeutik"]/child::FakName
```

```
<FakName>Philosophie</FakName>
```

# Verkürzte Syntax

- . Aktueller Referenzknoten
- .. Vaterknoten
- / Abgrenzung einzelner Schritte oder Wurzel
- ElemName1/ElemName2/ElemName3
- // descendant-or-self::node()
- @AttrName Attributzugriff

# XPath: Beispiele

```
doc("Uni.xml")/Universität/Fakultäten/  
    Fakultät[FakName="Physik"]//Vorlesung
```

```
doc("Uni.xml")/Universität/Fakultäten/  
    Fakultät[position()=2]//Vorlesung
```

```
doc("Uni.xml")/Universität/Fakultäten/Fakultät[2]//Vorlesung
```

```
doc("Uni.xml")/Universität/Fakultäten/Fakultät[FakName="Physik"]/  
    ProfessorIn/Vorlesungen/Vorlesung
```

# Beispiel-Pfadausdrücke

```
document("uni.xml")/Fakultäten/Fakultät[FakName="Physik"]  
//Vorlesung
```

```
document("uni.xml")/Fakultäten/Fakultät[2]//Vorlesung
```

```
document("uni.xml")/Fakultäten/Fakultät[FakName="Physik"]/  
ProfessorIn/Vorlesungen/Vorlesung
```

```
document("Stammbaum.xml")/Person[Name="Kain"]  
/@Vater->/Name
```

```
document("uni.xml")//Vorlesung[Titel="Mäeutik"]/  
@Voraussetzungen->/Titel
```

# XML-Anfragsyntax

- FLWOR-Ausdrücke
  - For ..
  - Let ...
  - Where ...
  - Order by ...
  - Return ...

# XML-Beispielanfrage

```
<Vorlesungsverzeichnis>
```

```
{for $v in doc("Uni.xml")//Vorlesung
```

```
return
```

```
$v}
```

```
</Vorlesungsverzeichnis>
```

```
<VorlesungsVerzeichnis>
```

```
<Vorlesung VorlNr=„V5022“>
```

```
<Titel>Glaube und Wissen</Titel>
```

```
<SWS>2</SWS>
```

```
</Vorlesung>
```

```
...
```

```
</VorlesungsVerzeichnis>
```



<Vorlesungsverzeichnis>  
 <Vorlesung VorlNr="V5022">  
 <Titel>Glaube und Wissen</Titel>  
 <SWS>2</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5001" VorlNr="V5041">  
 <Titel>Ethik</Titel>  
 <SWS>4</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5001" VorlNr="V5049">  
 <Titel>Mäeutik</Titel>  
 <SWS>2</SWS>  
 </Vorlesung>  
 <Vorlesung VorlNr="V4052">  
 <Titel>Logik</Titel>  
 <SWS>4</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5001" VorlNr="V5043">  
 <Titel>Erkenntnistheorie</Titel>  
 <SWS>3</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5043 V5041" VorlNr="V5052">  
 <Titel>Wissenschaftstheorie</Titel>  
 <SWS>3</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5041" VorlNr="V5216">  
 <Titel>Bioethik</Titel>  
 <SWS>2</SWS>  
 </Vorlesung>  
 <Vorlesung Voraussetzungen="V5052" VorlNr="V5259">  
 <Titel>Der Wiener Kreis</Titel>

# XML-Beispielanfrage

<Vorlesungsverzeichnis>

```
{for $v in doc("Uni.xml")//Vorlesung[SWS=4]
```

```
return
```

```
  $v}
```

</Vorlesungsverzeichnis>

<Vorlesungsverzeichnis>

```
{for $v in doc("Uni.xml")//Vorlesung
```

```
  where $v/SWS = 4
```

```
return
```

```
  $v}
```

</Vorlesungsverzeichnis>



# XML-Beispielanfrage

```
<VorlesungsverzeichnisNachFakultät>
```

```
  {for $f in doc("Uni.xml")/Universität/Fakultäten/Fakultät
```

```
    return
```

```
      <Fakultät>
```

```
        <FakultätsName>{$f/FakName/text()}</FakultätsName>
```

```
        {for $v in $f/ProfessorIn/Vorlesungen/Vorlesung
```

```
          return $v}
```

```
      </Fakultät>}
```

```
</VorlesungsverzeichnisNachFakultät>
```

# Joins in XQuery

```
<MäeutikVoraussetzungen>
```

```
{for $m in doc("Uni.xml")//Vorlesung[Titel="Mäeutik"],  
  $v in doc("Uni.xml")//Vorlesung  
  where contains($m/@Voraussetzungen,$v/@VorlNr)  
  return $v/Titel}
```

```
</MäeutikVoraussetzungen>
```

```
<MäeutikVoraussetzungen>
```

```
<Titel>Grundzüge</Titel>
```

```
</MäeutikVoraussetzungen>
```

# XML-Beispielanfrage

**<ProfessorenStammbaum>**

**{for \$p in doc("Uni.xml")//ProfessorIn,**

**\$b in doc("Stammbaum.xml")//Person,**

**\$km in doc("Stammbaum.xml")//Person,**

**\$kv in doc("Stammbaum.xml")//Person**

**where \$p/Name = \$b/Name and \$km/@id = \$b/@Mutter and**

**\$kv/@id = \$b/@Vater**

**return**

**<ProfMutterVater>**

**<ProfName>{\$p/Name/text()}</ProfName>**

**<MutterName>{\$km/Name/text()}</MutterName>**

**<VaterName>{\$kv/Name/text()}</VaterName>**

**</ProfMutterVater> }**

**</ProfessorenStammbaum>**

# XML-Beispielanfrage

<GefährdetePersonen>

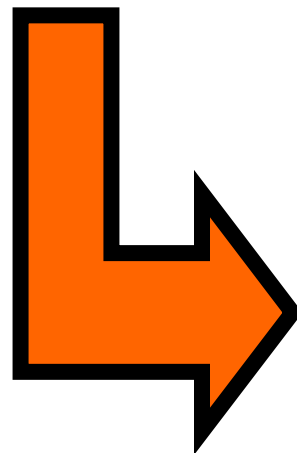
```
{for $p in doc("Stammbaum.xml")//Person[Name = "Kain"],
```

```
  $g in doc("Stammbaum.xml")//Person[
```

```
    @Vater = $p/@Vater and @Mutter = $p/@Mutter]
```

```
  return $g/Name }
```

</GefährdetePersonen>



```
<GefährdetePersonen>  
  <Name>Kain</Name>  
  <Name>Abel</Name>  
</GefährdetePersonen>
```

# XML-Beispielanfrage

```
<Professoren>
```

```
  {for $p in doc("Uni.xml")//ProfessorIn
```

```
    let $v := $p/Vorlesungen/Vorlesung
```

```
    where count($v) > 1
```

```
    order by sum($v/SWS)
```

```
    return
```

```
      <ProfessorIn>
```

```
        {$p/Name}
```

```
        <Lehrbelastung>{sum($v/SWS)}</Lehrbelastung>
```

```
      </ProfessorIn>
```

```
    }
```

```
</Professoren>
```

# XML-Beispielanfrage

**<Professoren>**

**<ProfessorIn>**

**<Name>Russel</Name>**

**<Lehrbelastung>8.0</Lehrbelastung>**

**</ProfessorIn>**

**<ProfessorIn>**

**<Name>Kant</Name>**

**<Lehrbelastung>8.0</Lehrbelastung>**

**</ProfessorIn>**

**<ProfessorIn>**

**<Name>Sokrates</Name>**

**<Lehrbelastung>10.0</Lehrbelastung>**

**</ProfessorIn>**

**</Professoren>**

# Leider kein Dereferenzierungsoperator

```
<VorlesungsBaum>
```

```
{for $p in doc("Uni.xml")//Vorlesung
```

```
return
```

```
  <Vorlesung Titel="{ $p/Titel/text() }">
```

```
    {for $s in doc("Uni.xml")//Vorlesung
```

```
      where contains($p/@Voraussetzungen,$s/@VorlNr)
```

```
      return <Vorlesung Titel="{ $s/Titel/text() }"> </Vorlesung> }
```

```
  </Vorlesung> }
```

```
</VorlesungsBaum>
```

# Ergebnis

<VorlesungsBaum>

<Vorlesung Titel="Glaube und Wissen"/>

<Vorlesung Titel="Ethik">

<Vorlesung Titel="Grundzüge"/>

</Vorlesung>

<Vorlesung Titel="Mäeutik">

<Vorlesung Titel="Grundzüge"/>

</Vorlesung>

<Vorlesung Titel="Logik"/>

<Vorlesung Titel="Erkenntnistheorie">

<Vorlesung Titel="Grundzüge"/>

</Vorlesung>

<Vorlesung Titel="Wissenschaftstheorie">

<Vorlesung Titel="Ethik"/>

<Vorlesung Titel="Erkenntnistheorie"/>

</Vorlesung>

<Vorlesung Titel="Bioethik">

<Vorlesung Titel="Ethik"/>

</Vorlesung>

<Vorlesung Titel="Der Wiener Kreis">

<Vorlesung Titel="Wissenschaftstheorie"/>



# Rekursion ... einfach

```
for $m in doc("Bauteile.xml")/Bauteil
    [Beschreibung="Maybach 620 Limousine"]
let $teile := $m//Bauteil
return
    <Kosten>
        <Verkaufspreis> {$m/@Preis} </Verkaufspreis>
        <PreisDerEinzelteile> {sum($teile/@Preis)}
        </PreisDerEinzelteile>
    </Kosten>
```

```
<Kosten>
    <Verkaufspreis Preis="350000"/>
    <PreisDerEinzelteile>59000.0</PreisDerEinzelteile>
</Kosten>
```

# Rekursion ... schwieriger

```
<!DOCTYPE VorlesungsBaum [  
  <!ELEMENT VorlesungsBaum (Vorlesung *)>  
  <!ELEMENT Vorlesung (Vorlesung *)>  
  <!ATTLIST Vorlesung  
    Titel CDATA #REQUIRED>  
>
```

# Rekursion ... schwieriger

```
declare function local:eineEbene($p as element()) as element()
{
  <Vorlesung Titel="{ $p/Titel/text() }">
    {
      for $s in doc("Uni.xml")//Vorlesung
      where contains($p/@Voraussetzungen,$s/@VorlNr)
      return local:eineEbene($s)
    }
  </Vorlesung>
};
```

```
<VorlesungsBaum>
{
  for $p in doc("Uni.xml")//Vorlesung
  return local:eineEbene($p)
}
</VorlesungsBaum>
```

# Ergebnis

```
<VorlesungsBaum>
  <Vorlesung Titel="Glaube und Wissen"/>
  <Vorlesung Titel="Ethik">
    <Vorlesung Titel="Grundzüge"/>
  </Vorlesung>
  <Vorlesung Titel="Mäeutik">
    <Vorlesung Titel="Grundzüge"/>
  </Vorlesung>
  <Vorlesung Titel="Logik"/>
  <Vorlesung Titel="Erkenntnistheorie">
    <Vorlesung Titel="Grundzüge"/>
  </Vorlesung>
  <Vorlesung Titel="Wissenschaftstheorie">
    <Vorlesung Titel="Ethik">
      <Vorlesung Titel="Grundzüge"/>
    </Vorlesung>
    <Vorlesung Titel="Erkenntnistheorie">
      <Vorlesung Titel="Grundzüge"/>
    </Vorlesung>
  </Vorlesung>
  <Vorlesung Titel="Der Wiener Kreis">
    <Vorlesung Titel="Wissenschaftstheorie">
      <Vorlesung Titel="Ethik">
```

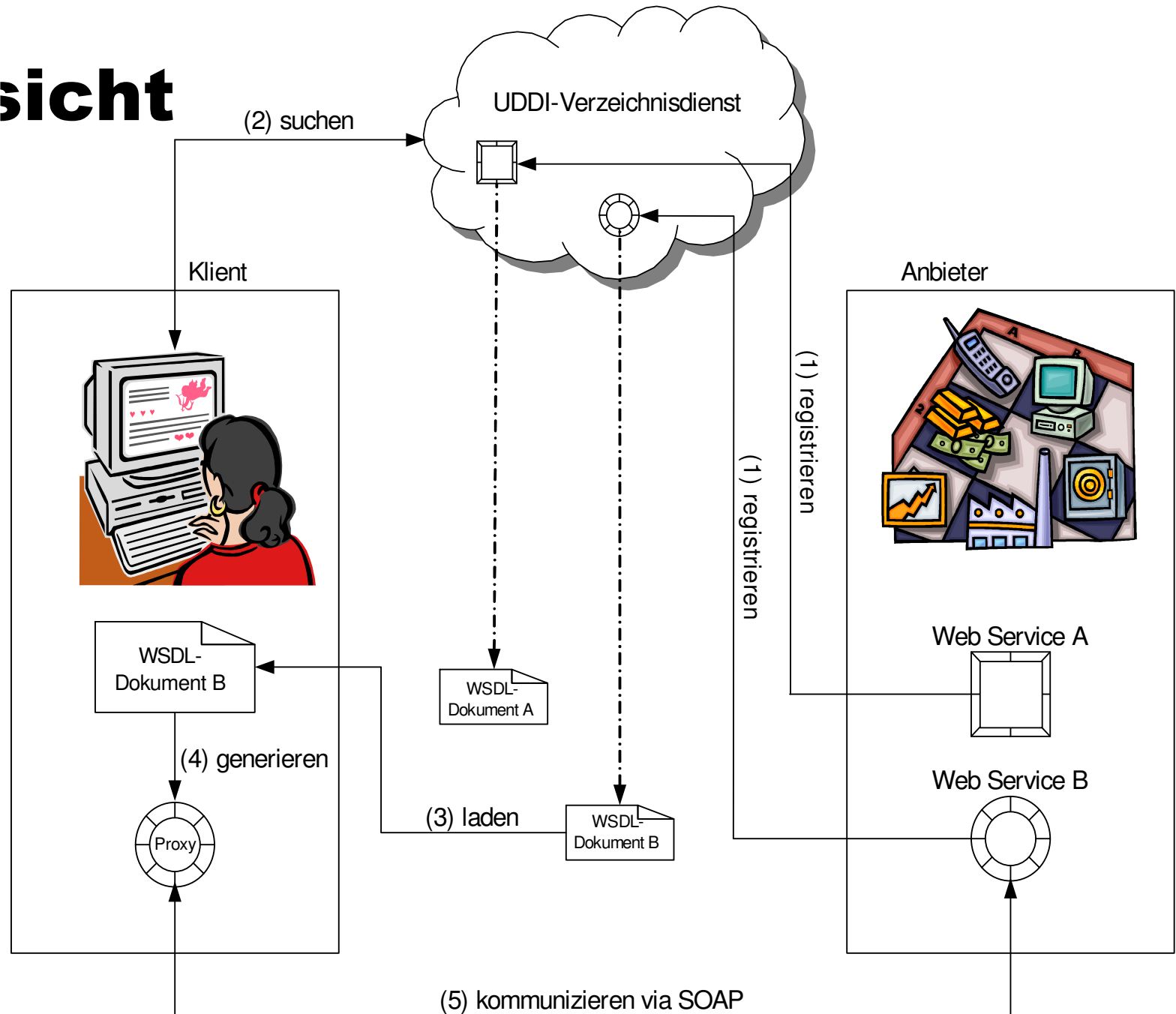
# Web-Services

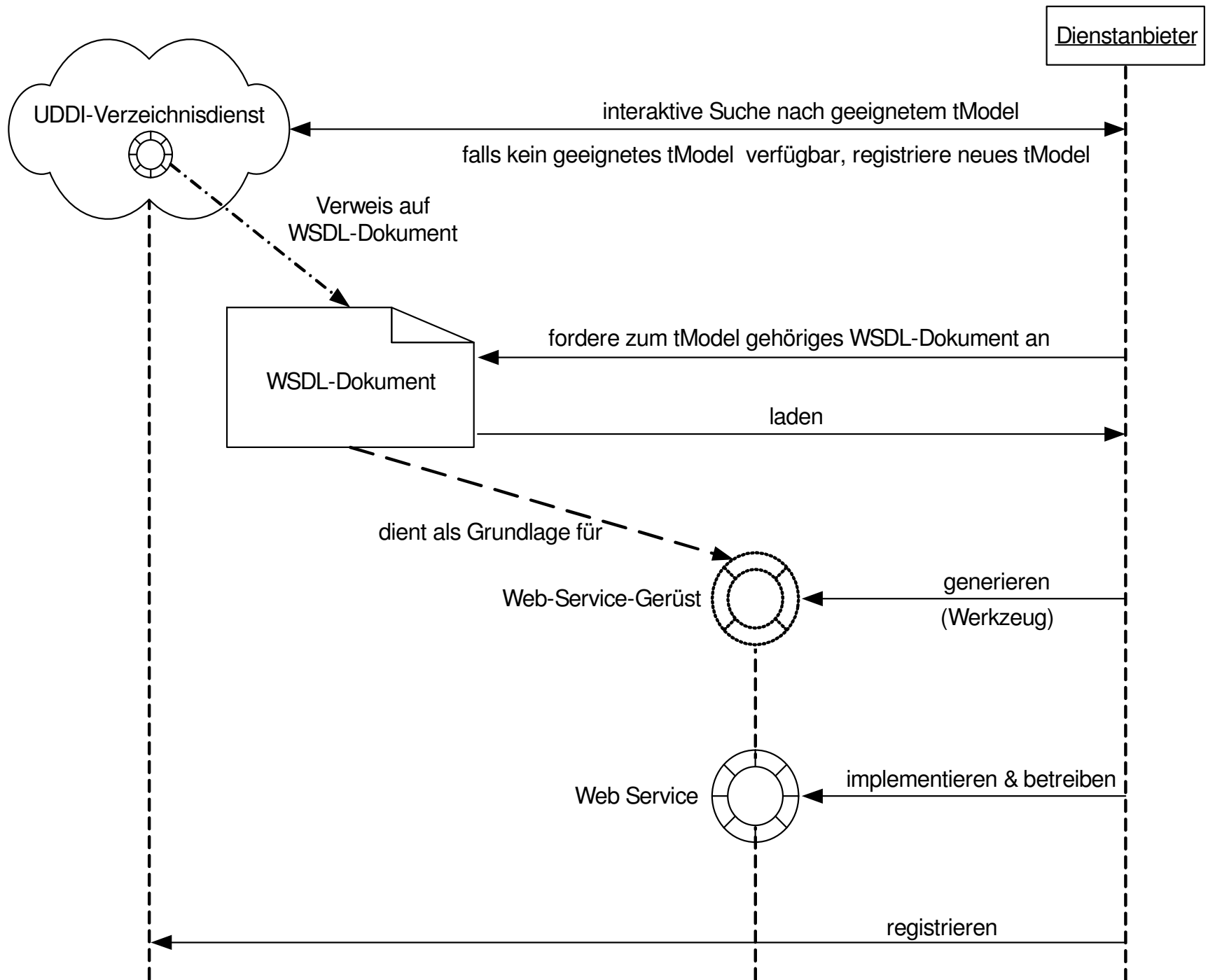
- XML wird die lingua franca des Internets
- Wird jetzt auch für die Kommunikation zwischen Programmen benutzt
- SOAP: Simple Object Access Protocol
  - Basiert auf XML
  - Ermöglicht i.w. entfernte Prozeduraufrufe

# Standards: die Wichtigsten

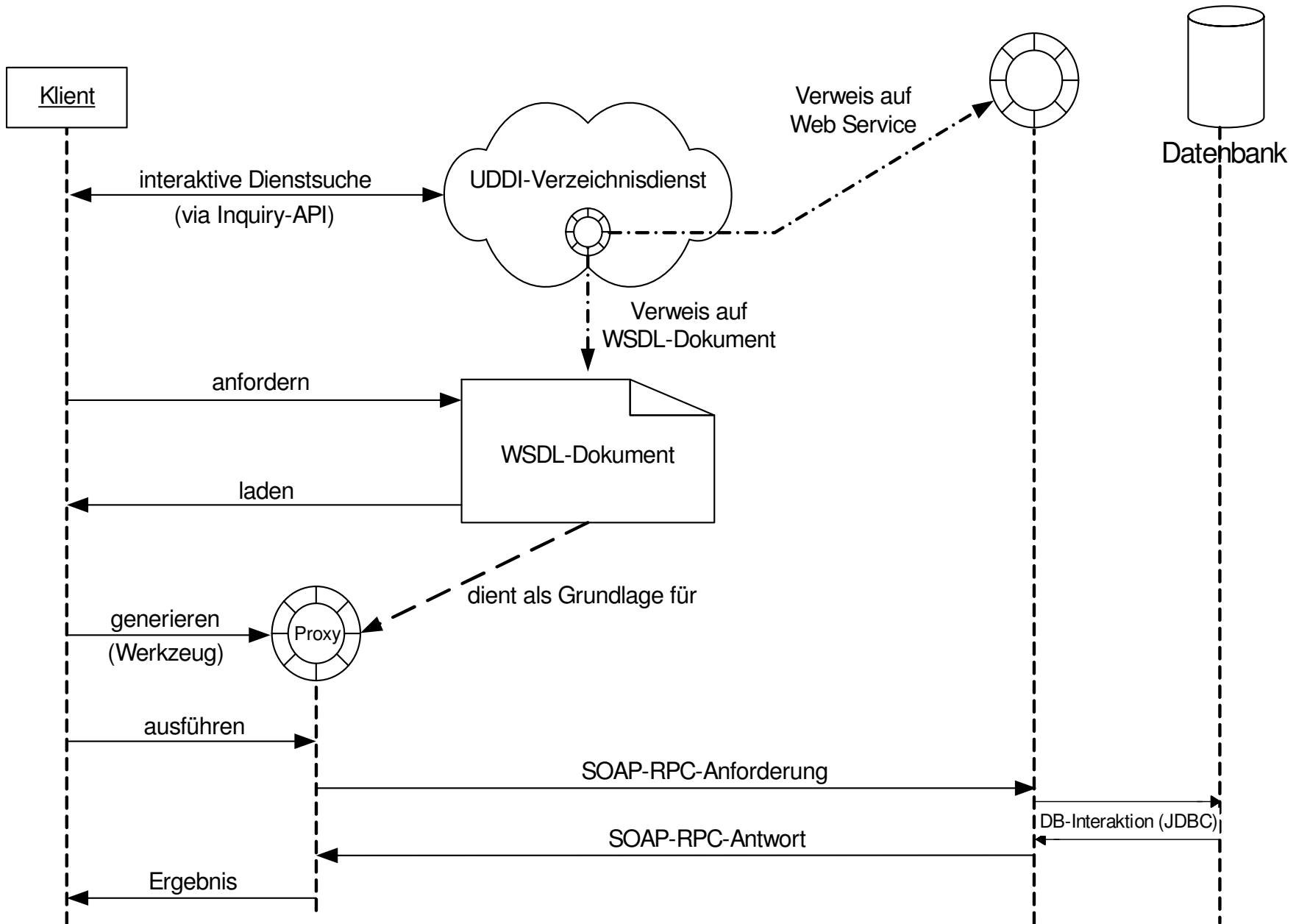
- SOAP (Simple Object Access Protocol von IBM, Microsoft, u.a.),
- UDDI (Universal Description, Discovery and Integration von HP, IBM, Intel, Microsoft, SAP, Software AG, Sun, u.a.),
- WSDL (Web-Services Description Language von Ariba, IBM und Microsoft),
- WSFL (Web-Services Flow Language von IBM),
- XLANG (Microsoft) und
- WS-Inspection (Web-Service Inspection Language von IBM und Microsoft).

# Übersicht









# SOAP-Kommunikation

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <soap:Body>
    <ns1:getLehrUmfangVonProfessor
      xmlns:ns1="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl">
      <ProfName xsi:type="xsd:string">Sokrates</ProfName>
    </ns1:getLehrUmfangVonProfessor>
  </soap:Body>
</soap:Envelope>

<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <soap:Body>
    <ns1:getLehrUmfangVonProfessorResponse
      xmlns:ns1="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl">
      <LehrUmfang xsi:type="xsd:int">10</LehrUmfang>
    </ns1:getLehrUmfangVonProfessorResponse>
  </soap:Body>
</soap:Envelope>
```

# WSDL: Web-Service Description Language

```
<?xml version="1.0" ?>
<definitions name="UniVerwaltung"
  targetNamespace="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl"
  xmlns:tns="http://www.db.fmi.uni-passau.de/UniVerwaltung.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetLehrUmfangVonProfessorRequest">
    <part name="ProfName" type="xsd:string"/>
  </message>
  <message name="GetLehrUmfangVonProfessorResponse">
    <part name="LehrUmfang" type="xsd:int"/>
  </message>

  <portType name="UniVerwaltungPortType">
    <operation name="getLehrUmfangVonProfessor">
      <input message="tns:GetLehrUmfangVonProfessorRequest"/>
      <output message="tns:GetLehrUmfangVonProfessorResponse"/>
    </operation>
  </portType>

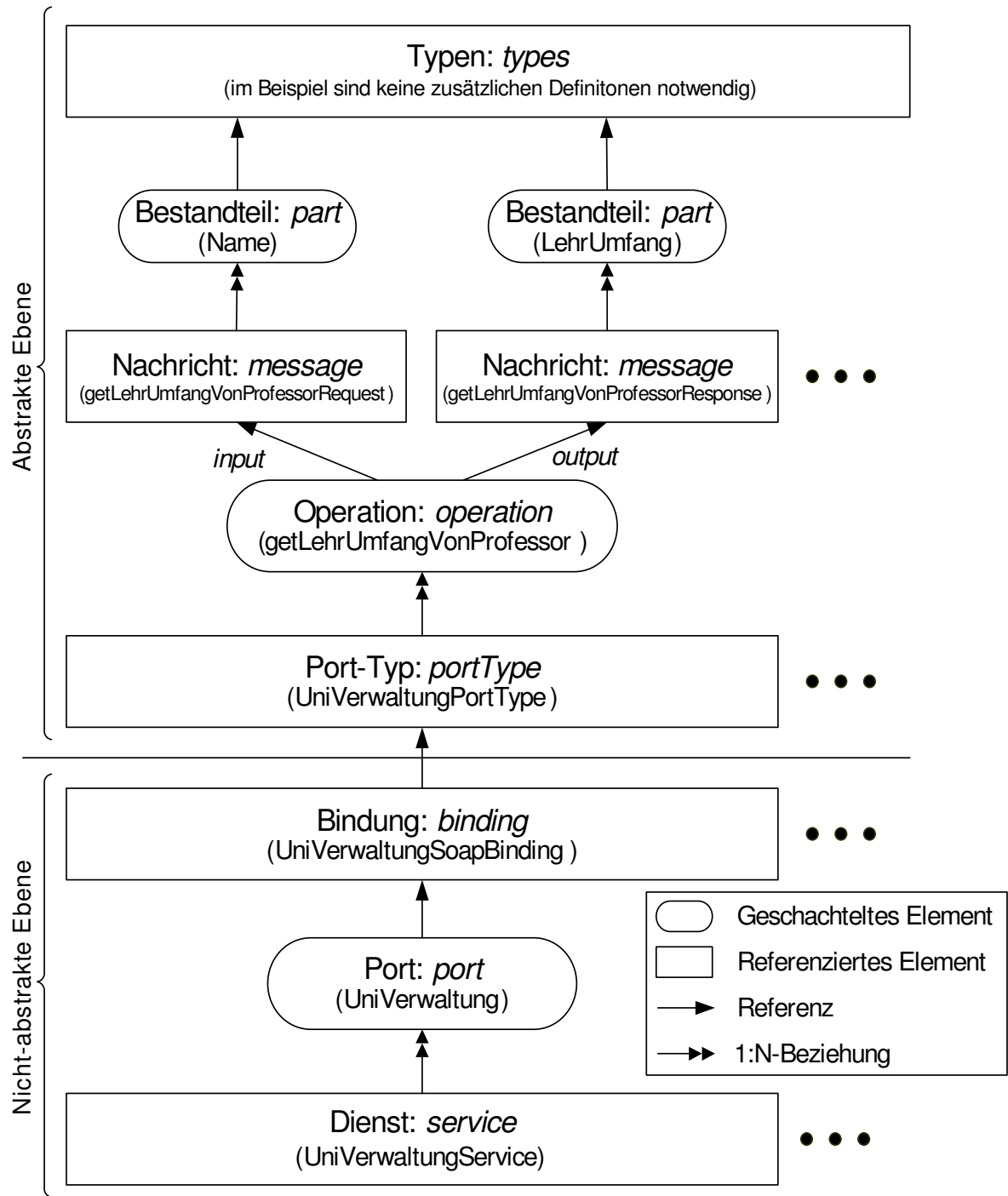
  ...
```

```
<binding name="UniVerwaltungSOAPBinding" type="tns:UniVerwaltungPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="getLehrUmfangVonProfessor">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded" namespace="UniVerwaltung"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="UniVerwaltung"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
</binding>

<service name="UniVerwaltungService">
  <port name="UniVerwaltung" binding="tns:UniVerwaltungSOAPBinding">
    <soap:address location=
      "http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung"/>
  </port>
</service>

</definitions>
```



# Implementierung des Web-Services

```
public class UniVerwaltungSOAPBindingImpl
    implements UniVerwaltung.UniVerwaltungPortType {
    public int getLehrUmfangVonProfessor(java.lang.String profName)
        throws java.rmi.RemoteException {
        return InquireDB.getLehrUmfangVonProfessor(profName); } }
```

```
import java.sql.*;
class InquireDB {
    public static int getLehrUmfangVonProfessor(String profName) {
        int LehrUmfang = 0;
        try { // connect to database:
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@devilray:1522:lsintern","WSUSER","Passwort");
            Statement stmt = conn.createStatement();
            ResultSet rset = stmt.executeQuery(
                "select sum(v.SWS) as LehrUmfang "
                + "from Vorlesungen v, Professoren p "
                + "where v.gelesenVon = p.PersNr and p.Name = '" + profName + "'");
            rset.next();
            LehrUmfang=java.lang.Integer.parseInt(rset.getString("LehrUmfang"));
            // disconnect
            rset.close(); stmt.close(); conn.close();
        } catch (Exception e) {}
        return LehrUmfang; } }
```

# Aufruf des Web-Services (Klient)

```
package UniVerwaltung;
import java.net.URL;

public class Klient {
    public static void main(String[] args) throws Exception {
        UniVerwaltungService uvws = new UniVerwaltungServiceLocator();
        UniVerwaltungPortType uv = uvws.getUniVerwaltung(new URL
            ("http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung"));
        System.out.println("Lehrumfang von Professor/in " +
            "Sokrates" + ": " +
            uv.getLehrUmfangVonProfessor("Sokrates")); //Dienstinvokation
    }
}
```

# Handgestrickter Klient

```
import java.io.*; import java.net.*;
```

```
public class ClientUniVerwaltung {  
    private static final int BUFF_SIZE = 100;
```

```
    public static void main(String[] argv) throws Exception {
```

```
        String request =
```

```
        "<?xml version='1.0' encoding='UTF-8'?>" +
```

```
        "<soap:Envelope " +
```

```
            "xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' " +
```

```
            "xmlns:xsd='http://www.w3.org/2001/XMLSchema' " +
```

```
            "xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' " +
```

```
            "soap:encodingStyle= " +
```

```
                "'http://schemas.xmlsoap.org/soap/encoding/'> " +
```

```
        "<soap:Body> " +
```

```
            "<ns1:getLehrUmfangVonProfessor " +
```

```
                "xmlns:ns1='http://www.db.fmi.uni-passau.de/' +
```

```
                "UniVerwaltung.wsdl'> " +
```

```
                    "<ProfName xsi:type='xsd:string'>Sokrates</ProfName>" +
```

```
                    "</ns1:getLehrUmfangVonProfessor>" +
```

```
        "</soap:Body>" +
```

```
        "</soap:Envelope>";
```



# Handgestrickter Klient ... cont'd

```
URL url = new URL(
    "http://www.db.fmi.uni-passau.de/axis/services/UniVerwaltung");
URLConnection conn = (URLConnection) url.openConnection();
```

```
conn.setDoOutput(true); conn.setUseCaches(false);
conn.setRequestProperty("Accept", "text/xml");
conn.setRequestProperty("Connection", "keep-alive");
conn.setRequestProperty("Content-Type", "text/xml");
conn.setRequestProperty(
    "Content-length",
    Integer.toString(request.length()));
conn.setRequestProperty("SOAPAction", "\" \");
```

```
OutputStream out = conn.getOutputStream();
out.write(request.getBytes()); out.flush();
```

```
StringBuffer response = new StringBuffer(BUFF_SIZE);
InputStreamReader in =
    new InputStreamReader(conn.getInputStream(), "UTF-8");
char buff[] = new char[BUFF_SIZE]; int n;
while ((n = in.read(buff, 0, BUFF_SIZE - 1)) > 0) {
    response.append(buff, 0, n);
}
out.close(); in.close();
System.out.println( response.toString() );
```

```
}
}
```