Inhaltsverzeichnis

1. XML

Dieses Thema wird in folgenden Dateien behandelt:

☑ xml-grundlagen.odt

☑ xml-fuers-www.odt

☑ xml-fuers-programmieren.odt

☑ xml-fuer-datenbanken.odt

1.1. Ziele

☑ Xml-grundlagen:
Behandelt werden die **Grundlagen** der eXtensible Markup Language.

☐ XML, DTD, XML-Schema

☑ xml-fuers-www:
XML fürs Web.

- ☐ **Transformierung** von XML-Dokumenten mittels Stylesheets (eXtensible Stylesheet Language XSL) und XSLT (Transformation)

☑ xml-fuers-programmieren:
XML fürs Programmieren

- ☐ Die **DOM-Schnittstelle** (Document Object Model) und SAX (Simple API for XML).

☑ Xml-fuer-datenbanken:

XML für Datenbanken.

- ☐ Mapping: RDBM – XML
- ☐ SQL/XML
- ☐ XQuery
- ☐ XML-Datenbanken

☑ Quellen:

- ☐ http://www.db.informatik.uni-kassel.de/?uri=Lehre/WS0405/XML/
- ☐ http://dret.net/lectures/xml/basics
- ☐ http://de.wikipedia.org/wiki/Extensible_Markup_Language
- ☐ http://xmlsoft.org/index.html
- ☐ http://xmlsoft.org/examples/index.html
- ☐ http://www.w3schools.com/
- ☐ http://de.selfhtml.org/xml/intro.htm

2. XML: Grundlagen

2.1. XML: Was ist das?

Weitere Information zu XML finden Sie unter:
http://www.w3.org/XML und http://www.w3.org/TR/xml

XML selbst und viele darauf aufbauende Standards wurden vom World Wide Web Consortium **W3C** verabschiedet.

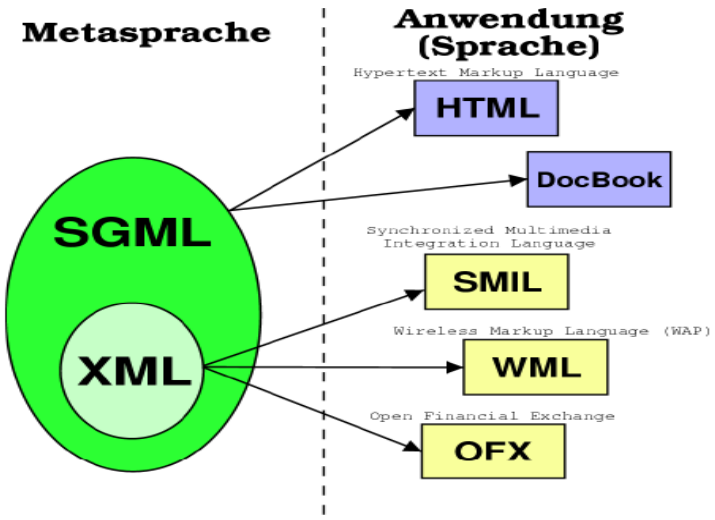
XML ist
keine Markup-Sprache (Beschreibungssprache), sondern eine **Metabeschreibungssprache**.
D. h. **eine Sprache für die Definition** anderer Markup-Sprachen wie etwa HTML.

Dabei gilt XML als Vereinfachung der Standard Generalized Markup Language (SGML).
XML wird HTML nicht ersetzen. Vielmehr ist das neuere XML-basierte XHTML eine „sauberere" Version von HTML.

2.1.1. Ein erstes Beispiel: book.xml

Book.xml

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

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

</bookstore>
```



2.1.2. Unterschiede zwischen XML- und HTML-Dokumenten

HTML	XML
Es gibt Tags ohne oder mit optionalem Ende-Tag: <P>	Jeder Tag muss durch einen Ende-Tag beendet werden: <P></P>
Attributwerte müssen nicht in Anführungszeichen stehen: <TD ALIGN=LEFT>	Attributwerte müssen immer in Anführungszeichen stehen: <TD ALIGN="LEFT">
Es gibt Attribute ohne Wert: <TABLE BORDER>	Jedes Attribut hat einen Wert
HTML ist nicht case-sensitive	XML ist case-sensitive

2.1.3. Verwendung von XML

Die Verwendung von XML liegt in verschiedenen Bereichen:

☑ Als erstes wird XML für die **Entwicklung** von verschiedenen **Beschreibungsprachen** (Mark-Up-Sprachen) verwendet. Vorhandene Beispiele dafür sind z. B. HTML, SVG und MathML.

☑ Der zweite Anwendungsbereich liegt in der **Definition** von **Austauschformaten** zwischen verschiedenen Anwendungen in heterogenen Netzen. Hierzu dienen die Standards SOAP und XML-RPCs.

Zudem gibt es folgende Vorteile:

☑ XML-Dokumente müssen eindeutige Regeln und Strukturen einhalten, sodass ihre **Verarbeitung durch Anwendungsprogramme einfacher und sicherer** geschehen kann als z. B. die Verarbeitung von HTML-Dokumenten.

☑ **Inhalt und Darstellung** eines Dokuments werden voneinander **getrennt**, sodass unterschiedliche Darstellungen desselben Dokuments möglich sind. Somit können z. B. mehrere geräteabhängige Visualisierungen aus einem einzigen Dokument erzeugt werden.

☑ XML-Dokumente sind **einfache**, aber **wohl strukturierte Textdokumente**. Damit sind sie sowohl von menschlichen Lesern als auch von Anwendungsprogrammen einfach zu interpretieren und zu verarbeiten.

☑ EDI (Electronic Data Interchange) **Datenformat zur Kommunikation**. Anwendungen können Informationen austauschen.

XML ist selbstbeschreibend
Dadurch prädestiniert als Austauschformat

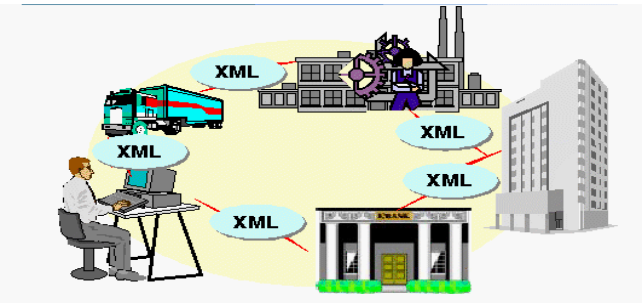
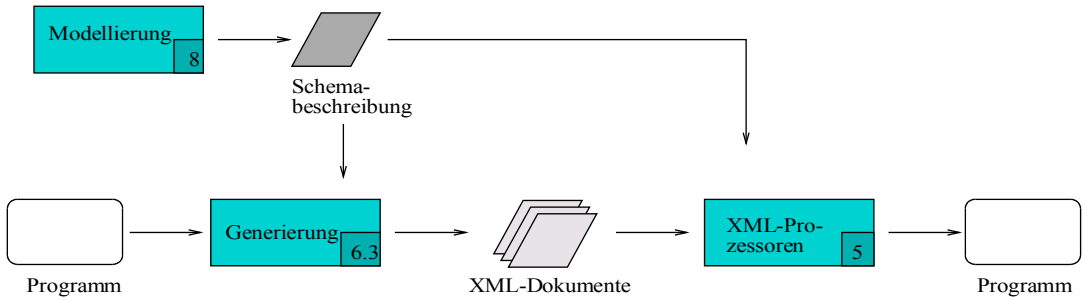


Abbildung von Roland Boendgen

☑ **Kommunikation von Programmen**



☑ XML wird nicht verwendet, wenn die Daten sehr einfach sind und einem starren Aufbau entsprechen (z.B. Tabellen), also klar und einfach strukturiert sind.

2.1.4. Überblick: Aufbau von XML Dokumenten

Eine XML Datei besteht aus 3 Teilen:

1. Prolog <?xml ..... ?>
2. Verweis/ bzw. Angaben zur Struktur der nachfolgenden Daten
3. Daten

Teilnehmerliste („teilnehmer.xml")

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

<!DOCTYPE TeilnehmerS SYSTEM "teilnehmer.dtd">

<TeilnehmerS vorlesung="Einführung in XML">

     <Teilnehmer matrNr="4711">
          <name>Wiese</name>
          <vorname>hans</vorname>
          <semester>6</semester>
          <fachb>16</fachb>
     </Teilnehmer>
     <Teilnehmer matrNr="4722">
          <name>Wald</name>
          <vorname>Max</vorname>
          <semester>8</semester>
```

```
            <fachb>17</fachb>
      </Teilnehmer>

</TeilnehmerS>
```

Dieses Dokument besteht, wie jedes XML-Dokument überhaupt, aus Inhalten, die mit Markup-Symbolen durchsetzt sind.

Tags
Die spitzen Klammern(<>) und die von ihnen umschlossenen Namen heißen tags. Wie man sieht, unterscheidet sich der End-Tag vom Start-Tag durch den Schrägstrich („/“) als erstes Zeichen nach der öffnenden spitzen Klammer.
Tags ohne Inhalt, werden durch <tagname /> markiert.

Elemente
Tags begrenzen und bezeichnen Teile des Dokuments, die wir Elemente nennen, und fügen weitere Informationen hinzu, die zur Definition der Struktur beitragen.

Inhalt
Der Text zwischen den Tags ist der Inhalt (Content) des Elements, zusammen sind sie der Inhalt des Dokuments, also die nackte Information, wie der Inhalt eines Briefes oder der Name des Absenders.

Prolog
Die Anweisung **<?xml version="1.0" encoding="UTF-8"?>** in der ersten Zeile ist eine sog. XML-Deklaration (XML Processing Instruction). Sie teilt einem XML-Prozessor mit, dass dieses Dokument ein XML-Dokument ist. Zusätzlich wird die verwendete XML-Version 1.0 und der Zeichensatz UTF-8 angegeben.

Die Tags <TeilnehmerS> und <Teilnehmer> enthalten weitere Unterelemente. Sie dienen also der Strukturierung des Dokuments. Das name-Element dagegen enthält einen Text als Inhalt.

Attribute
Die Angaben matrNr="..." und version="..." sind sog. Attribute.
Ein Element kann mehrere Attribute enthalten, die weitere Informationen über das Element angeben. Die Attributwerte sind in Hochkommata (einfache oder doppelte) eingeschlossen. In unserem Beispiel gibt z. B. Das Attribut matrNr die Matrikelnummer eines Teilnehmers an.

Namespaces
Namespaces sind Namensräume, vergleichbar in etwa mit den Packages in Java. Damit bestimmte Bezeichner in verschiedenen Zusammenhängen mit unterschiedlicher Bedeutung benutzt werden können, macht XML sehr ausgiebig Gebrauch vom Konzept der Namespaces, was XML-Dokumente leider manchmal sehr unübersichtlich machen kann.

Siehe auch:
http://www.w3.org/TR/REC-xml-names

2.2. Aufbau von XML-Dokumenten im Detail

XML-Dokumente haben den folgenden Grundaufbau:

```
<?xml version='1.0' ...?>
<!DOCTYPE ...>
<rootElement attr="Wert" ...>
     <firstElement attr="Wert">
          Inhalt von firstElement
     </firstElement>
     <lastElement attr="Wert">
          Inhalt von lastElement
     </lastElement>
</rootElement>
```

Wir gehen im folgenden auf die einzelnen Teile dieses Dokuments ein.

2.2.1. Die XML-Deklaration

Die erste Zeile ist die sog. XML-Deklaration. Sie muss in jedem XMLDokument enthalten sein.

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

Sie kann die folgenden Attribute haben:
■ **version**
Spezifiziert die verwendete XML-Version im Dokument.

■ **encoding**
Definiert die im Dokument verwendete Zeichenkodierung. Einige verbreitete Zeichenkodierungen sind: US-ASCII, ISO-8859-1, UTF-8 und UTF-16. Falls encoding nicht angegeben wird, wird die Standard-Zeichenkodierung UTF-8 verwendet.

2.2.2. Die Dokumenttyp-Deklaration

Eine DTD (Document Type Definition) muss deklariert werden. Die Instruktion kann gegenwärtig die zwei Formen annehmen:

```
<!DOCTYPE root-element PUBLIC "name" "URL-of-DTD">

<!DOCTYPE root-element SYSTEM "URL-of-DTD">
```

Mit PUBLIC werden (in Zukunft!) öffentlich zugängliche DTDs spezifiziert. Diese sollen als registrierter Name in einem internen oder externen Repositorium bekannt sein. Schlägt die Ermittlung über name fehl, wirdauf die folgende URL zugegriffen. Alternativ kann man sofort auf privat verfügbare DTDs zugreifen, dies geht mit der Variante SYSTEM.

Die Angabe name bei PUBLIC benutzt den sog. Formal Public Identifier (FPI). Diese Namenskonvention benutzt einen Namen mit vier Segmenten, die durch Doppelschrägstriche (slashes) getrennt sind, z. B.:

"-//O`Reilly//DTD//EN"

Das Minus deutet auf einen unregistrierten Namen hin, der ggf. nicht eindeutig ist. Ein Plus (+)

wäre z. B. ein beim W3C registrierter Eintrag. Das zweite Segment ist der Autor oder die Organisation, die den Namen veröffentlicht, das dritte Segment die Inhaltsform (hier DTD). Zuletzt folgt der Sprachcode, hier EN für Englisch.

2.2.3. XML-Elemente

Elemente sind die Grundbausteine von XML-Dokumenten. In XML werden Elemente durch Tags in spitzen Klammern in der Form <elementname> ... </element-name> eingeklammert.

2.2.4. Der Elementname

☑ muss mit einem Buchstaben oder Unterstrich beginnen und
☑ kann eine beliebige Anzahl von Buchstaben, Zahlen, Bindestrichen, Punkten und Unterstrichen enthalten.

Das Doppelpunktzeichen (:) wird als Trenner für Namensräume verwendet (später in diesem Kapitel).

Leerzeichen, Tabulatoren, Zeilenumbrüche, Gleichheitzeichen und Anführungszeichen sind Separatoren und dürfen deshalb in Element- und Attributnamen nicht verwendet werden.

Sonderzeichen wie & oder ? dürfen ebenfalls nicht in Elementnamen enthalten sein.

Zwischen der ersten spitzen Klammer und dem Elementnamen darf kein Leerzeichen stehen (z. B. < Buch>), dagegen können beliebige Trennzeichen nach dem Elementnamen, und zwischen den Attributen stehen.

Damit kann man ein Element auf mehreren Zeilen verteilen:

```
<Buch
ISBN="X-3-89721-286-2" titel='Einführung in XML'>
...
</Buch>
```

Da XML Unicode als Zeichensatz verwendet, können Elementnamen in einer beliebigen, von Unicode unterstützten Sprache geschrieben werden.

Beispiele für gültige Elementnamen in den Start-Tags sind

```
<_>, <_a>, <b4711>, <VON-BIS>, <Name.Vorname>.
```

Beispiele für nicht gültige Elementnamen in Tags sind etwa

```
<4711>, <.punkt>, <a&b>, <verstanden?>.
```

2.2.5. Start- und Endtags

Ein Element besteht aus einem Start-Tag, einem Inhalt und einem End-Tag. Der Inhalt kann aus reinem Text bestehen, aus weiteren Unterelementen oder einer Mischung aus beidem. Die Ende-Markierung (End-Tag) wird durch </elementName> angegeben, und darf nicht, wie in HTML, weggelassen werden.

Ist der Inhalt leer (leeres Element), so kann man Start- und End-Tag zu einem Tag zusammenziehen indem man ein '/'-
Zeichen an das Ende des Tags vor die schließende spitze Klammer
schreibt:

```
<leeresElement attr1='...' attrn='...' />
```

Der Start- und End-Tag eines Elements müssen sich beide im gleichen Eltern-Element befinden. So ist z. B.

```
<a>So <b>geht es</a> nicht</b>
```

nicht erlaubt.

Da das Kleinerzeichen (<) für den Beginn eines Tags steht, darf es nicht direkt im Text verwendet werden. Dazu können die vordefinierten Entities für < (<) und für > (>) verwendet werden.

Sehr wichtig bei der Texteingabe ist darauf zu achten, dass XML alle Leerzeichen, Zeilenumbrüche und Tabulatorzeichen beibehält.
Dies ist bei Programmiersprachen und anderen Auszeichnungssprachen, wo Leerzeichen in der Regel ignoriert werden (siehe xml:space weiter unten), anders.

2.2.6. Attribute

Mit Attributen kann man Elementen zusätzliche oder genauere Informationen hinzufügen. Man kann z. B. Elementen eindeutige Bezeichner durch ein Attribut geben, oder eine Eigenschaft über dieses Element beschreiben (Beispielsweise das href-Attribut für das A-Element in HTML).

Eine Attributangabe besteht aus einem Attributnamen, einem Gleichheitszeichen und einem Wert in Anführungszeichen. Dabei können einfache oder doppelte Anführungszeichen verwendet werden:

```
<person id="123">max Mustermann</person>  oder
<person id='123'>max Mustermann</person>
```

Für Attributnamen gelten die gleichen Regeln wie für Elementnamen. Ein Element darf ein Attribut nur einmal enthalten.

```
<x a="wert 1" a="wert 2"/> ist nicht erlaubt.
```

Ein Element kann eine beliebige Anzahl von Attributen haben, solange jedes über einen eindeutigen Namen verfügt.

2.2.7. Reservierte Attribute

Die folgenden Attributnamen sind im XML-Standard mit einer festen Bedeutung reserviert und dürfen deshalb nicht neu definiert werden:

■ **xml:lang**
Mit diesem Attribut wird die Sprache des Elements angegeben. Dies ist nützlich für die

Erstellung von mehrsprachigen Dokumenten. Die Angabe verwendet einen zweibuchstabigen Sprachcode (Kleinschreibung wie de, en ist üblich). Ein weiterer Qualifier kann verwendet werden, um eine Sprach-Variante wie en-US zu spezifizieren. Vereinbarungsgemäß besteht dieser Qualifier aus zwei Großbuchstaben.

■ **xml:link**
Signalisiert einem XLink-Prozessor, dass ein Element ein Link-Element ist. XLink wird in späteren Kapiteln behandelt.

2.2.8. Wohlgeformte XML-Dokumente

Im Laufe der Entwicklung von HTML haben sich gewisse tolerierte Vereinfachungen und Unsauberkeiten eingeschlichen. So darf man bei Tabellen das schließende Element eines Tabellenfeldes </TD> weglassen, allerdings auf die Gefahr hin, dass das letzte Feld nicht richtig zugeordnet wird.

Dies ist in XML nicht zulässig. Alle Elemente müssen paarweise auftreten und "wohlgeschachtelt" (properly nested) sein, außer natürlich bei leeren Elementen wie etwa <picture src='...' />.

Damit ist

```
<Abschnitt>
     <Para>
          Blablabla
     <Para>
     noch mehr bla
</Abschnitt>
```

ungültiges XML (es fehlen jeweils die </Para> Endmarken).

Genauso wird

```
<b><i>Achtung: gekauft wie gesehen</b></i>
```

zwar in HTML akzeptiert, in XML aber zurückgewiesen.

Argumente von Attributen müssen in XML immer in Hochkommata eingeschlossen werden. HTML Browser erlauben auch das Weglassen, sofern der Wert keine Leerstellen enthält. Es sind sowohl einzelne als auch doppelte Hochkommata erlaubt.

Zuletzt sei darauf hingewiesen, dass HTML bekanntlich nicht auf Groß- und Kleinschreibung achtet, XML dagegen case sensitive ist!

<h2>Überschrift</H2> wird deshalb als XML-Dokument zurückgewiesen, ginge in HTML aber durch.

Weiterhin ist zu beachten:
■ Ein XML-Dokument besitzt ein eindeutiges Wurzelelement.

Dokumente, die korrekt geschachtelt sind und die obigen Punkte beachten, sind wohlgeformt (well-formed). Genügt ein Dokument auch einer zugehörigen DTD, dann nennt der Standard es gültig (valid).

2.2.9. Prozessor-Instruktionen

Prozessor-Instruktionen (PI, Processing Instructions) sind dazu gedacht, Anweisungen an externe Anwendungen in einem XML-Dokument einzubauen.

Diese werden nur von der adressierten Anwendung verstanden und interpretiert. Sie werden von anderen Anwendungen ignoriert.
Eine PI sieht wie folgt aus
<?anwendung attr1='Wert 1' ...?>

Als Beispiel habe wir schon die XML-Instruktion
<?xml version=... ?>
kennengelernt.

Ein weiteres Beispiel ist der Hinweis für die Verwendung eines Stylesheets:

```
<?xml-stylesheet type="text/xsl" href="filename.xsl"?>
```

Dies teilt einem Stylesheet-Prozessor mit, das XML-Dokument anhand des Stylesheets filename.xsl zu transformieren. Weitere Erklärungen: s. u.

2.2.10. Kommentare

Kommentare werden in XML durch <!-- ... --> angegeben. In einem Kommentar können beliebige Texte vorkommen. Innerhalb eines Kommentars dürfen aber keine doppelten Bindestriche (hyphens) enthalten sein.
Kommentare dürfen überall im Dokument stehen, nur nicht vor der XML-Deklaration und innerhalb von Tags. Sie werden von einem XMLProzessor ignoriert.

2.2.11. CDATA-Abschnitte

Mit dieser sog. Character-Data Sektion (nicht zu verwechseln mit der PCDATA-Angabe in DTDs) lassen sich Zeichenfolgen angeben, die der Parser nicht interpretiert. Die Folge muss mit <![CDATA[anfangen und mit]]> aufhören. Als Beispiel könnte man schreiben

```
<![CDATA[ In diesem XML-Kapitel besprechen wir
,,Deklarationen'', z. B. <?xml ...?> und <!DOCTYPE ...>,
CDATA[...]-Sektionen & vieles mehr
]]>
```

Andererseits sollte man innerhalb einer die CDATA-Sektion keine Entity-Referenzen wie &, <, >, " und ' verwenden, da diese nicht aufgelöst werden.

2.2.12. Namensräume

In XML-Dokumenten können Elemente aus verschiedenen Sprachen verwendet werden. Sie können z. B. eine mathematische Formel als MathML-Element in einem HTML-Dokument einbauen. Damit der XML-Prozessor (HTML-Browser) zwischen den Elementen beider Sprachen (HTML, MATHML) unterscheiden kann und diese unterschiedlich behandelt (z. B. Das Plugin für die Darstellung von MathML aufruft), wurden in XML Namensräume eingeführt.

Bevor ein Namensraum verwendet werden kann, muss dieser zuerst im Dokument deklariert werden. Die Deklaration erfolgt in der Form eines Attributes innerhalb eines Elements:

```
<namensraum:elementName xmlns:namensraum="url">
```

Man kann in einem Dokument mehrere Namensräume verwenden. Alle Nachkommen dieses Elements werden Teil des Namensraumes. Zur Verwendung von Namensräumen werden sie durch ein Namensraum-Präfix gefolgt von einem Doppelpunkt gefolgt vom lokalen Namen des Elements angegeben.

Der Wert des xmlns:-Attributs ist eine URL, die gewöhnlich zu der Organisation gehört, die den Namensraum unterhält. Es ist jedoch nicht erforderlich, dass der XML-Prozessor etwas mit dieser URL tut. Die Angabe der URL dient lediglich der eindeutigen Bezeichnung des Namensraums.

Im folgenden Beispiel werden die Namensräume vorlesung und skript deklariert:

```
<?xml version="1.0"  ?>
<vorlesung:beispiel
     xmlns:vorlesung=<"http://www.informatik.uni-kassel.de/~wegner/">

     <vorlesung:title>Namensraum</vorlesung:title>
     <vorlesung:code>4711-9999-XYZ</vorlesung:code>
     <vorlesung:Anfang>
          Auch Zwerge haben mal klein angefangen
     </vorlesung:Anfang>

     <skript:inhaltsverz
          xmlns:skript='http://www.informatik.uni-kassel.de/skript/'>
          <skript:kap nr='1' titel='Einführung'>
               <skript:abschnitt titel='Historie'/>
               <skript:abschnitt titel='Erste Schritte'/>
          </skript:kap>
     </skript:inhaltsverz>
</vorlesung:beispiel>
```

Wir können einen Namensraum als Standard-Namensraum deklarieren, indem wir den Doppelpunkt (:) und den Namen im xmlns-Attribut weglassen.

In unserem Beispiel können wir z. B. vorlesung als Standard-Namensraum benutzen:

```
<?xml version="1.0" ?>
<beispiel
     xmlns="http://www.informatik.uni-kassel.de/~wegner/">

     <title>Namensraum</title>
     <code>4711-9999-XYZ</code>
     <Anfang>
          Auch Zwerge haben mal klein angefangen
     </Anfang>
     <skript:inhaltsverz
          xmlns:skript='http://www.informatik.uni-kassel.de/skript/'>
```

```
          <skript:kap nr='1' titel='Einführung'>
          <skript:abschnitt titel='Historie'/>
          <skript:abschnitt titel='Erste Schritte'/>
          </skript:kap>
     </skript:inhaltsverz>
</beispiel>
```

2.3. DTD,XSD: Wohlgeformtheit, Validierung von XML-Dateien

Soll die syntaktische Korrektheit von XML-Dokumenten beurteilt werden, so wird zwischen zwei Stufen unterschieden:

☑ **well-formed (wohlgeformt)** ist ein Dokument, das:
- ☐ der XML-Syntax genügt
- ☐ genau ein Wurzelelement hat
- ☐ korrekt geschachtelt ist

Die Entscheidung, ob ein Dokument well-formed ist, kann also ohne die DTD getroffen werden.

☑ **valid** (**gültig**) ist ein Dokument, das:
- ☐ well-formed ist und
- ☐ über eine DTD verfügt
- ☐ dieser DTD entspricht

Zur Definition der **Struktur (Elemente und Regeln)** eines XML-Dokumentes wurde zuerst die sog. Data Type Definition (DTD) von SGML übernommen.

2.3.1. DTD: Ein Adressbuchbeispiel: wohlgeformt und gültig

Teilnehmerliste („teilnehmer.xml")

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

<!DOCTYPE TeilnehmerS SYSTEM "teilnehmer.dtd">

<TeilnehmerS vorlesung="Einführung in XML">

     <Teilnehmer matrNr="4711">
          <name>Wiese</name>
          <vorname>hans</vorname>
          <semester>6</semester>
          <fachb>16</fachb>
     </Teilnehmer>
     <Teilnehmer matrNr="4722">
          <name>Wald</name>
          <vorname>Max</vorname>
          <semester>8</semester>
          <fachb>17</fachb>
```

```
     </Teilnehmer>

</TeilnehmerS>
```

Das XML-Dokument ist wohlgeformt
Das obige XML-Dokument konnte vom Stylesheet-Prozessor bearbeitet und mithilfe eines Stylesheets in HTML übertragen werden. Es erfüllt also bestimmte Grundvoraussetzungen eines XML-Dokuments. Wir sagen, es ist wohlgeformt. D.h. die Start- und Endtags sind richtig gesetzt, usw.

Das XML-Dokument ist validiert
Doch welche Elemente kann man in einem solchen Dokument verwenden und welche Attribute haben sie? Das XML Dokument entspricht einer Grammatik.

Zur Festlegung dieser Grammatik dienen sowohl DTDs als auch XML Schemata. Wir betrachten hier eine einfache DTD zur Definition der obigen Sprache.

DTD für Teilnehmerlisten („teilnehmer0.dtd")

```
<!ELEMENT TeilnehmerS (Teilnehmer*)>
<!ATTLIST TeilnehmerS vorlesung CDATA #REQUIRED>
<!ELEMENT Teilnehmer (name, vorname, semester, fachb)>
<!ATTLIST Teilnehmer matrNr NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT semester (#PCDATA)>
<!ELEMENT fachb (#PCDATA)>
```

Zur Definition eines Elements dient die Deklaration

```
<!ELEMENT name (definition)>
```

Die Deklaration eines Attributes geschieht mit

```
<!ATTLIST elementname attributname definition>
```

Elemente mit Textinhalt werden mit dem Schlüsselwort (**#PCDATA**) angegeben (z. B. Name).

Elemente, die weitere Unterelemente enthalten, werden durch die Auflistung der erlaubten Unterelemente, getrennt durch Kommata, definiert (z. B. Teilnehmer).

Im XML-Dokument kann dann ein Verweis auf die verwendete DTD angegeben werden. Das geschieht wie folgt:

```
<?xml version='1.0' standalone='no'?>

<!DOCTYPE TeilnehmerS SYSTEM "teilnehmer0.dtd">

<TeilnehmerS>
...
</TeilnehmerS>
```

Bei kurzen DTDs bietet es sich an, diese direkt in das XML-Dokument wie folgt einzubauen:

```
<?xml version='1.0' standalone='yes'?>
<!DOCTYPE TeilnehmerS [
<!ELEMENT TeilnehmerS (Teilnehmer*)>
...
]>
<TeilnehmerS>
...
</TeilnehmerS>
```

Hier nun ein weiteres Beispiel:

Datei: ab.dtd

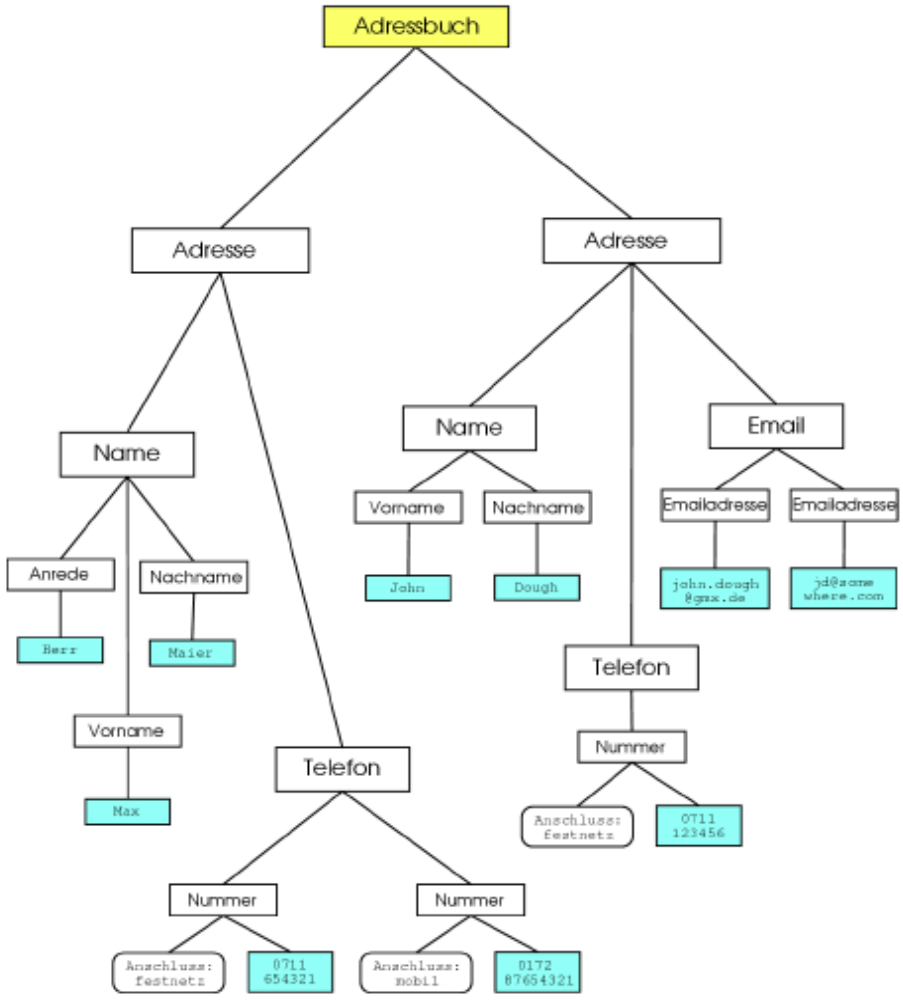
```
<!ELEMENT Adressbuch (Adresse)*>
<!ELEMENT Adresse (Name, Telefon, (Email)?)>
<!ELEMENT Name ((Anrede)?, (Vorname)?, Nachname)>
<!ELEMENT Anrede (#PCDATA)>
<!ELEMENT Vorname (#PCDATA)>
<!ELEMENT Nachname (#PCDATA)>
<!ELEMENT Telefon (Nummer)+>
<!ELEMENT Nummer (#PCDATA)>
<!ATTLIST Nummer
     Anschluss (mobil|festnetz) #REQUIRED>
<!ELEMENT Email (Emailadresse)+ >
<!ELEMENT Emailadresse (#PCDATA)>
```

Datei: ab.xml

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

<!DOCTYPE Adressbuch SYSTEM "ab.dtd">

<Adressbuch>

<Adresse>
<Name>
<Anrede>Herr</Anrede>
<Vorname>Max</Vorname>
<Nachname>Maier</Nachname>
</Name>
<Telefon>
<Nummer Anschluss="festnetz ">0711654321</Nummer>
<Nummer Anschluss="mobil">017287654321</Nummer>
</Telefon>
</Adresse>

<Adresse>
<Name>
<Vorname>John</Vorname>
<Nachname>Dough</Nachname>
</Name>
<Telefon>
```

```
<Nummer Anschluss="festnetz">07 11123456</Nummer>
</Telefon>
<Email>
<Emailadresse>john.dough@gmx.de</Emailadresse>
<Emailadresse>jd@somewhere.com</Emailadresse>
</Email>
</Adresse>

</Adressbuch>
```

Das XML Dokument als Baum



Bei kurzen DTDs bietet es sich an, diese direkt in das XML-Dokument wie folgt einzubauen:

```
<?xml version='1.0' standalone='yes'?>

<!DOCTYPE TeilnehmerS [
<!ELEMENT TeilnehmerS (Teilnehmer*)>
...
```

```
]>

<TeilnehmerS>
...
</TeilnehmerS>
```

2.3.2. XSD Dateien: XML-Schema

Da DTDs einige Nachteile aufweisen, wurde ein neuer XMLbasierter Standard für die Modellierung eines Dokuments entwickelt: **XML Schema** Language.
Siehe auch:
http://www.w3.org/XML/Schema

Charakteristika von XML Schema:

☑ XML-Schema beschreiben die Struktur eines XML-Dokuments (wie DTD).
☑ XML-Schema basieren auf XML-Syntax und können entsprechend geparst werden.
☑ XML-Schema unterstützen Datentypen wie z.B.
  ☐ integer
  ☐ string
  ☐ boolean
☑ In XML-Schema können eigene Datentypen definiert werden.

☑ XML-Schemas erlauben Gruppen von Elementen,
  indem alle Elemente genau z.B. einmal enthalten sein müssen (all), die Elemente mindestens einmal, aber auch öfter auftreten können (sequence) oder einer Auswahl aus mehreren Möglichkeiten (choice).

Beispiel: Ein xml-Schema

book.xsd

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

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
     elementFormDefault="qualified">

<xsd:element name="Title" type="xsd:string"/>
<xsd:element name="Author" type="xsd:string"/>
<xsd:element name="Date" type="xsd:string"/>
<xsd:element name="ISBN" type="xsd:string"/>
<xsd:element name="Publisher" type="xsd:string"/>

<xsd:element name="Book">
     <xsd:complexType>
     <xsd:sequence>
          <xsd:element ref="Title"/>
          <xsd:element ref="Author"/>
     <xsd:element ref="Date"/>
```

```
      <xsd:element ref="ISBN"/>
      <xsd:element ref="Publisher"/>
      </xsd:sequence>
      </xsd:complexType>
</xsd:element>

<xsd:element name="BookCatalogue">
      <xsd:complexType>
      <xsd:sequence>
      <xsd:element ref="Book" maxOccurs="unbounded"/>
      </xsd:sequence>
      </xsd:complexType>
</xsd:element>

</xsd:schema>
```

book.xml mit Verweis auf book.xsd

```
<?xml version="1.0"?>

<BookCatalogue xmlns="http://www.ppedv.de/schema/BookCatalogue"
      xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://http://URL/book.xsd">

<Book>
<Title>Computernetzwerke</Title>
<Author>Tanenbaum</Author>
<Date>1999</Date>
<ISBN>3-8272-9568-8</ISBN>
<Publisher>Prentice Hall</Publisher>
</Book>
<Book>
<Title>1984</Title>
<Author>George Orwell</Author>
<Date>1995</Date>
<ISBN>3-548-22562-4</ISBN>
<Publisher>Ullstein</Publisher>
</Book>
</BookCatalogue>
```

2.3.3. Markup Validation Service

siehe http://validator.w3.org/

2.4. Zusammenfassung: Uni Beispiel

2.4.1. Dozenten.dtd

```
<?xml version='1.0' encoding="ISO-8859-1" ?>
<!ELEMENT dozenten    (dozent*)>
<!ELEMENT dozent      (PersNr,Name,Rang,Raum,Amt*)>
<!ELEMENT PersNr      (#PCDATA)>
<!ELEMENT Name        (#PCDATA)>
<!ELEMENT Rang        (#PCDATA)>
<!ELEMENT Raum        (#PCDATA)>
<!ELEMENT Amt         (Bezeichnung,Termin?)>
<!ELEMENT Bezeichnung (#PCDATA)>
<!ELEMENT Termin      (#PCDATA)>
```

2.4.2. Dozenten.xml

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

<!DOCTYPE dozenten SYSTEM "dozenten.dtd" >

<dozenten>
  <dozent>
    <PersNr>2125</PersNr>
    <Name>Sokrates</Name>
    <Rang>C4</Rang>
    <Raum>226</Raum>
    <Amt>
      <Bezeichnung>Dekan</Bezeichnung>
      <Termin>vormittags</Termin>
    </Amt>
  </dozent>
  <dozent>
    <PersNr>2126</PersNr>
    <Name>Russel</Name>
    <Rang>C4</Rang>
    <Raum>232</Raum>
  </dozent>
  <dozent>
    <PersNr>2127</PersNr>
    <Name>Kopernikus</Name>
    <Rang>C3</Rang>
    <Raum>310</Raum>
    <Amt>
      <Bezeichnung>Weltraumbeauftragter</Bezeichnung>
    </Amt>
    <Amt>
      <Bezeichnung>Studienberater</Bezeichnung>
      <Termin>donnerstags 10-12 Uhr</Termin>
    </Amt>
  </dozent>
  <dozent>
```

```
    <PersNr>2133</PersNr>
    <Name>Popper</Name>
    <Rang>C3</Rang>
    <Raum>52</Raum>
  </dozent>
  <dozent>
    <PersNr>2134</PersNr>
    <Name>Augustinus</Name>
    <Rang>C3</Rang>
    <Raum>309</Raum>
  </dozent>
</dozenten>
```

Diese Datei kann man mit http://www.stg.brown.edu/service/xmlvalid/ validieren.

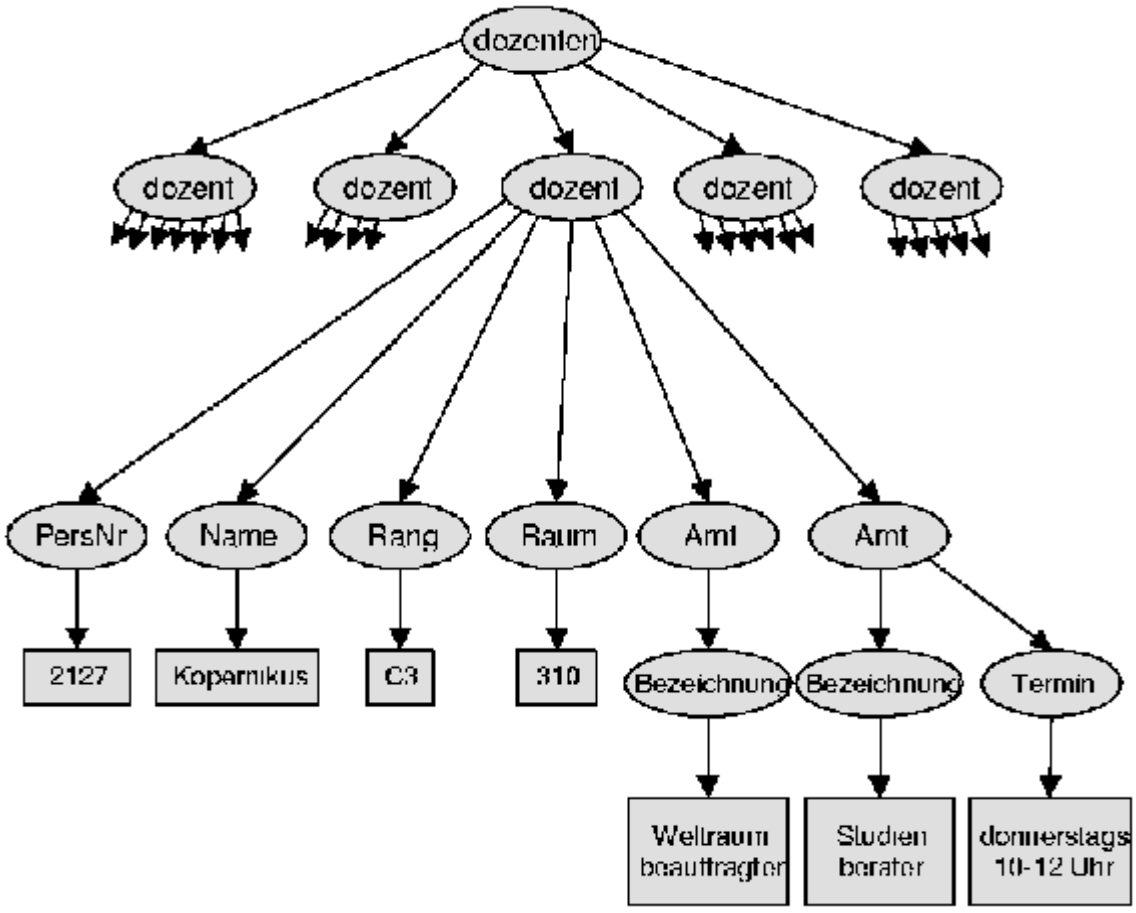
Nun noch das XML-Schema

2.4.3. Dozenten.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="dozenten">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="dozent" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="dozent">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="PersNr"/>
                <xs:element ref="Name"/>
                <xs:element ref="Rang"/>
                <xs:element ref="Raum"/>
                <xs:element ref="Amt" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="PersNr" type="xs:string"/>

    <xs:element name="Name" type="xs:string"/>

    <xs:element name="Rang" type="xs:string"/>

    <xs:element name="Raum" type="xs:string"/>

    <xs:element name="Amt">
```

```
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Bezeichnung"/>
                <xs:element ref="Termin" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="Bezeichnung" type="xs:string"/>

    <xs:element name="Termin" type="xs:string"/>

</xs:schema>
```

2.4.4. Das dazugehörige DOM



Aufgabenverzeichnis