Proseminar SoSe 2006

Intelligent Information Retrieval in WWW

Cristina Vertan
vertan@informatik.uni-hamburg.de

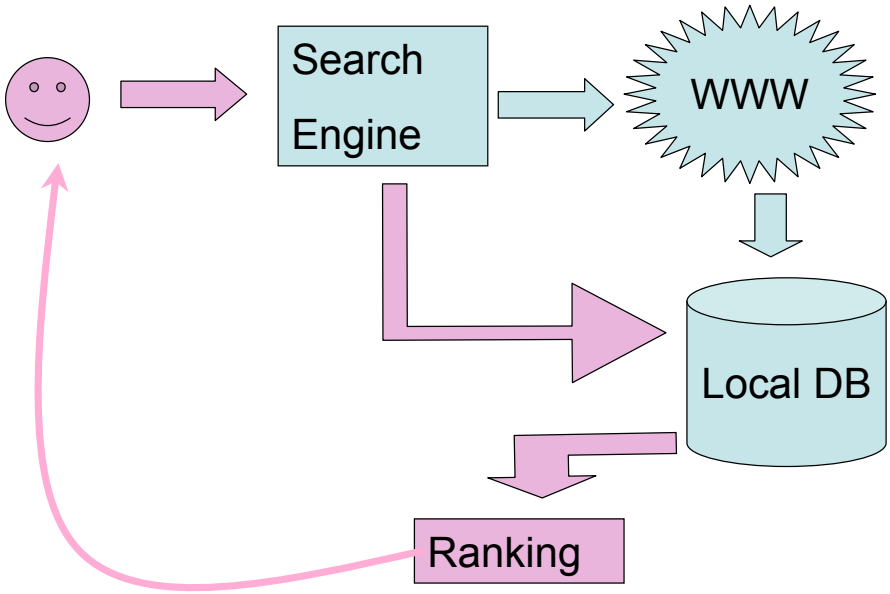
Inhalt

- Suchmechanismen
- WWW und Semantic Web
- Sprachen für Web und Semantic Web

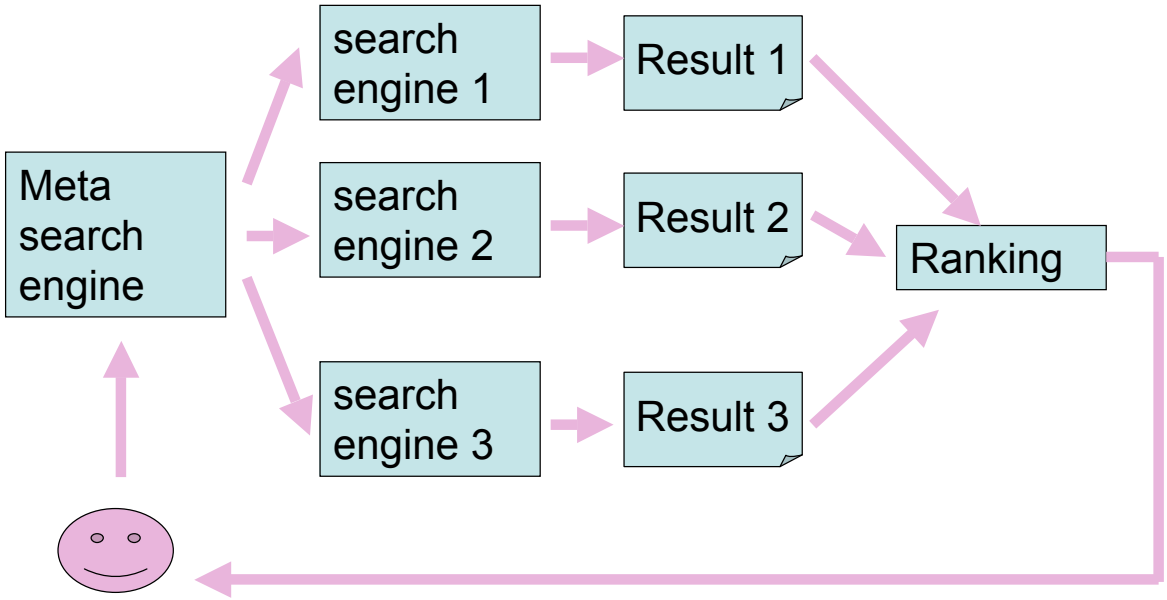
Typen von Suchverfahren

- Manuel (man folgt eventuel Hierachien die von der Author der Seite zur Verfügung gestellt wurden)
- Algoritmisch
  - Universele Verfahren (keine Begrenzung in Bezug auf Domäne, Sprache, geographische Lage)
  - Verfahren für speziele Domäne , Sprachen oder Dateitypen
  - Archivierungsverfahren (die Ergebnise werden auf der Speicherplatte gespeichert)

Automatisce Suchverfahren



Meta Search Engines

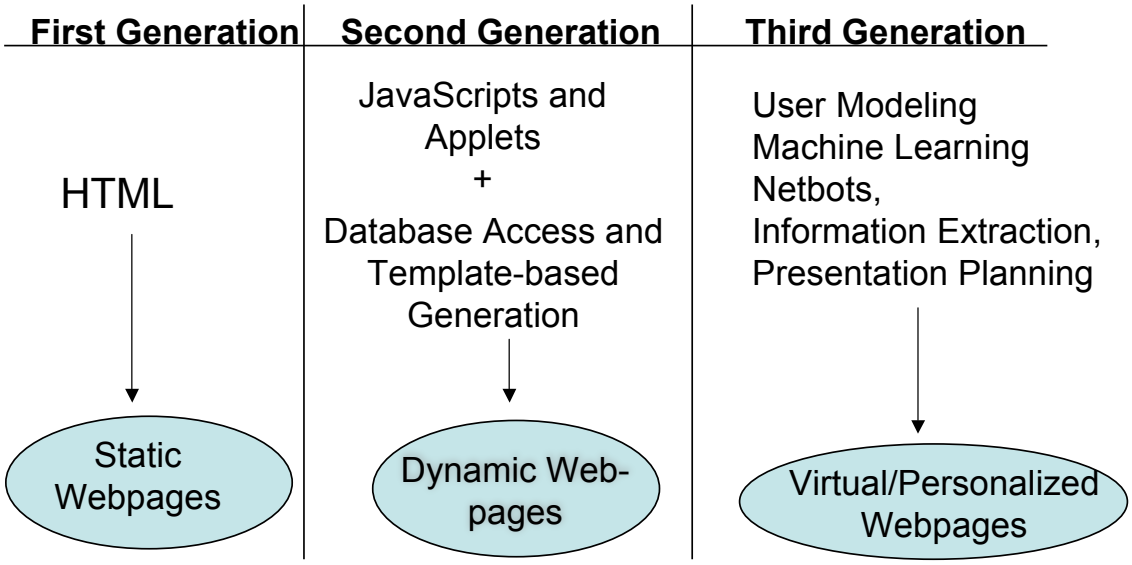


Aktuelle und zukünftige Perspektiven für WWW

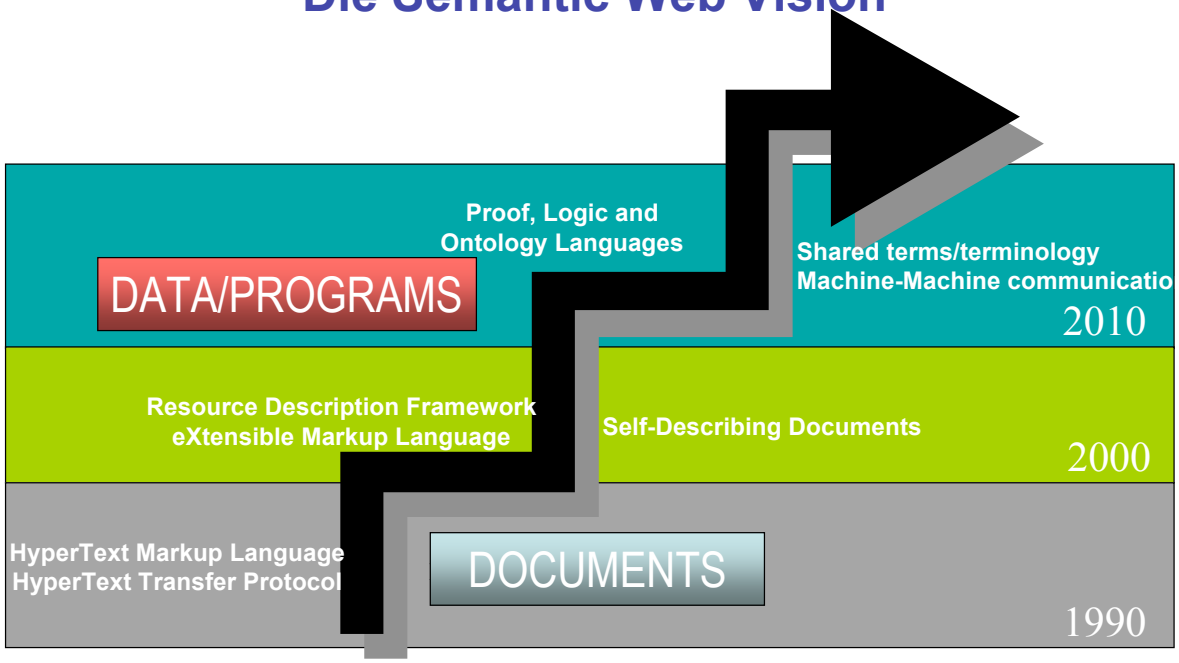
Aktuelle Funktionalität des Webs

- Das WWW wurde für die menschliche Benutzung entworfen.
- Das Webinhalt enthält keine strukturelle Information. (z.B. die Informationen die von Datenbanken extrahiert werden, enthalten nach der Extraktion keine Information über ihre ursprüngliche Zuordnung in der Datenbank)
- Typische Operationen in WWW:
  - Suche
  - Kommunikation
- Die Suchwerkzeuge (AltaVista, Yahoo, Google) sind meistens auf Stichwort-Suche eingestellt.

Drei Generationen von Web seiten



Die Semantic Web Vision

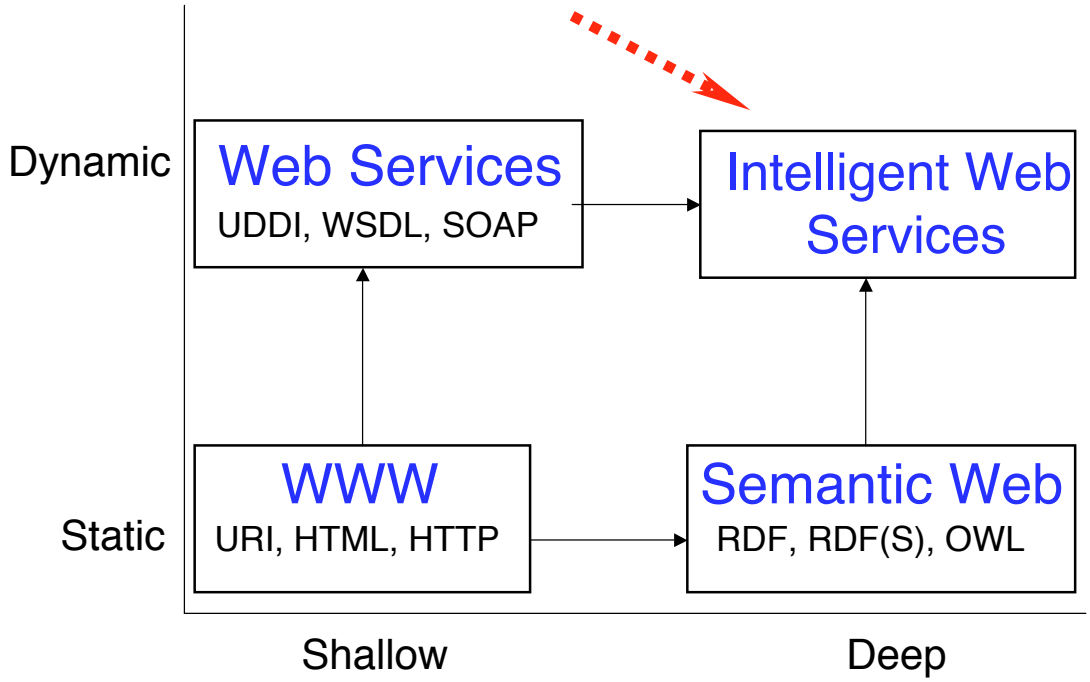


© Berners-Lee, Hendler; *Nature*, 2001

Integration von Semantic Web und Webdienste

© D. Fensel

Bringing the web to its full potential



Probleme der Suchwerkzeuge im aktuellen WWW

- **„High recall, low precision“:** Die nötigen Webseiten werden gefunden , aber zusammen mit anderen 28 758 nicht so relevanten oder total unnötigen Dokumenten.
  - Z.B. suche nach „*Java programming language*“ wird auch alle Dokumenten über die Insel Java , eventuell die dortige Sprache herausfinden
- **„Low or no recall“:** manchmal wichtige Seiten werden nicht gefunden. Das kann beobachtet werden wenn man dieselbe Anfrage an zwei oder drei Suchmaschinen gibt. Oft sind die Ergebnisse unterschiedlich.
- **Große Terminologie- und Sprachabhängigkeit**. Wichtige Informationen, die Synonyme oder Übersetzungen der Anfragewörter enthalten werden nicht gefunden
- **Ergebnisse sind einzelne Webseiten**. Wenn das Ergebnis in mehrere Dokumenten verstreut ist, muss man mehre Anfragen starten und dann die partiellen Informationen aggregieren.

Warum entstehen Probleme während der WWW-Suche?

- Hauptproblem: die Bedeutung des Webinhaltes ist nicht für automatische Prozesse verfügbar.
- Für eine Maschine ist es z.B schwer zu unterscheiden zwischen:
  - *Das Buch ist eine gute Quelle für .....*

  Und
  - *Das Buch wäre eine gute Quelle für ...wenn nicht...*
- Textverstehen-Methoden (Sprachverarbeitung, KI) können zur Zeit keine Lösungen, die domäne- und sprachunabhängig sind, liefern.
- Lösung: Eine neue Methode für Datenannotation und Inferenzmechanismen im WWW

Definitionen des Semantic Webs

- Semantic = Bedeutung
- Semantic Web = Datenbedeutung die auch maschinell gefunden und bearbeitet sein kann
- Es gibt mehrere Sichtweisen,was „SemanticWeb“ ist:
  - „*Die Sicht von maschinell lesbaren Daten*“: Die Daten sind so dargestellt dass sie von den Computern interpretierbar sind, und das nicht nur für Darstellungszweck. (Berners-Lee)
  - „*Die Sicht von Intelligenten Agenten*“ . Das aktuelle Web soll maschinell lesbar, so dass intelligente Agenten Daten finden und manipulieren können.
  - „*Dies verteilte Datenbanksicht*“. Das Semantic Web wird für die Daten sein was das Web für Menschen ist. D.h. Semantic Web wird einen einhaltlichten Mechanismus für Speicherung und Durchsuchung der Daten bereitstellen. (W3C)

Von Web zu Semantic Web -Anwendungen-

- Es gibt bereits mehrere Anwendungsgebiete die direkt vom Semantic Web profitieren können:
  - Wissensmanagement
  - eBusiness
  - eLearning
  - Personal intelligent agents
  - Sprachverarbeitung

Technologien des Semantic Webs

Wie entsteht das Semantic Web?

- Semantic Web wird „on top“ des aktuellen Webs gebaut.
- Kein revolutionärer wissenschaftlicher Fortschritt ist nötig.
- Partielle Lösungen zu allen Teilen sind bereits vorhanden.
- Heutige Herausforderungen:
  - Integration
  - Standardisierung
  - Werkzeugentwicklung
  - Benutzerakzeptanz.
- Folgende Technologien spielen eine Hauptrolle in der Implementierung des Semantic Webs:
  - Explizite Metadaten
  - Ontologien
  - Logik (Inferenzregeln)
  - Agenten

Explizite Metadaten -1-

- Der Webinhalt ist für Menschen und nicht für die Maschinen formatiert.
- Standard Repräsentationssprache ist HTML:
- Beispiel: Beschreibung eines Beratungsbüros:

```
<h1> Willkomen in unserem Zentrum .. </h1>
...Für allgemeine Informationen rufen Sie bitte unsere
   Sekretärin ...
Für fachliche Fragen rufen sie bitte Dr. ....
<h2> Sprechstunden </h2>
Mo - Do 11 - 17 Uhr
Fr 8 - 11.30 Uhr
Während der Ferienzeit  in <a href=„...“>Hamburg </a> wird
   unser Zentrum geschlossen.
```

Wie unterscheidet die Maschine zwischen Sekretärin und Dr.?

Wie wird automatisch die Öffnungszeit gerechnet?

Explizite Metadaten -2-

- Ersetzung von HTML durch Metadatensprachen, die den Inhalt repräsentieren können.
- Metadaten = Daten über Daten
- Für das o.g. Beispiel:

```
<zentrum>
   <dienst> Beratung</dienst>
   <zentrumName> ..... </zentrumName>
   <staff>
      <fachlicheBeratung> Dr. .... </fachlicheBeratung>
      <sekretariat> ..... </sekretariat>
   </staff>
</zentrum>
```

- Repräsentationssprache: **XML**:Teil der Datenbedeutung ist mit Metadaten darstellbar.

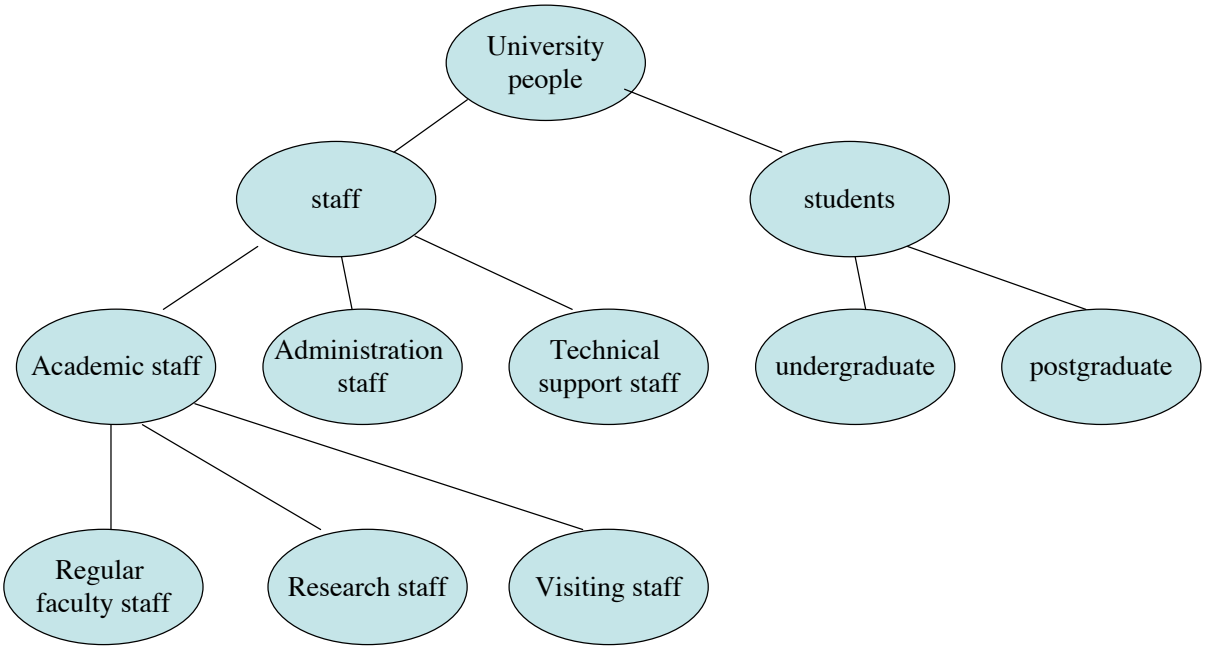
Ontologien -1-

- In der Informatik:„Eine Ontologie ist eine explizite und formale Repräsentation einer Konzeptualisierung.“ (R. Struder).
- Eine Ontologie beschreibt formal ein Diskursdomäne und enthält:
  - Begrenzte Liste von Konzepten (Objektklassen) und
  - Beziehungen zwischen diesen Konzepten
    - Vererbung (isSubclassOf)
    - Merkmale (isRelatedWith, isSimilarWith)
    - Wertbegrenzungen (z.B. Sprechstunden hat nur das wissenschaftliche Personal)
    - „disjoint statements“ (*fachliche Staff* und *administrative Staff* sind disjoint)
    - Logische Beziehungen zwischen Objekten (ein Zentrum muss mindestens 5 fachliche Berater haben)

Ontologien -2-

- Die Metadaten in XML dargestellt werden, werden in Ontologien organisiert. Insofern lassen sich mögliche Überlappungen vermeiden
- Z.B. in einer Institution könnte <staff> nur das fachliche Personal sein, in einer anderen das gesamte Personal.
- Das Problem wird durch eine Ableitung an der entsprechenden Ontologie gelöst.
- Die Organisation von Konzepten in einer Ontologie, sowie die Ableitung von Metadaten auf die Ontologie, ermöglicht auch die Definition von Inferenzregeln.

Ontologie -Beispiel



Logik

- Logik beschäftigt sich mit Denkprinzipien:
  - Formale Sprachen für Wissensrepräsentation
  - Bedeutung von Äußerungen (deklaratives Wissen), d.h. beschrieben wird: **was** entsteht und nicht **wie** entsteht(es).
  - Inferenzregeln die implizite Wissen in explizite Wissen umwandeln.

z.B.:wenn wir wissen :

```
Prof(X)→faculty(X)
faculty(X) →staff(X)
Prof(michael)
```

Können wir schliessen, daß,:

```
faculty(michael)
Staff(michael)
Prof(X) →staff(X)
```

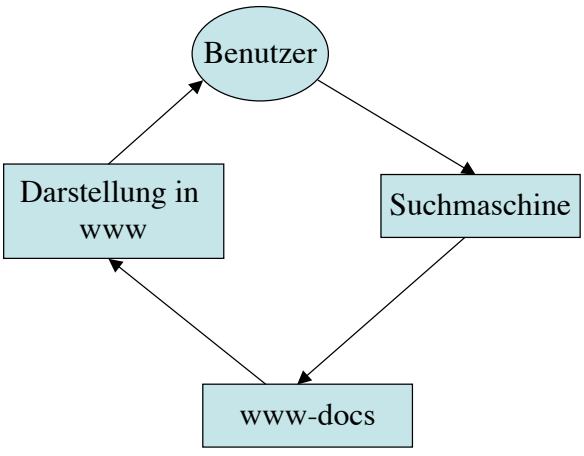
Das Wissen wird normalerweise aus Ontologien extrahiert.

Agenten -1-

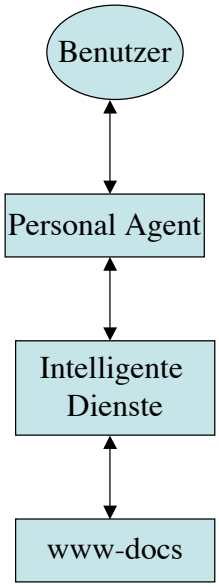
- Agenten sind Softwarekomponenten die autonom arbeiten.
- Z.B.Ein Agent im Semantic Web:
  - bekommt Aufgaben und Preferenzen vom Benutzer,
  - sucht Informationen in Web-Quellen,
  - kommuniziert mit anderen Agenten,
  - vergleicht Informationen über Benutzeranforderungen und -präferenzen,
  - wählt einige Optionen aus und
  - antwortet dem Benutzer.

Agenten -2-

Heute



Zukunft



Agenten und Semantic Web

- Metadaten werden für die Identifizierung und die Informationextraktion von Webquellen benutzt.
- Ontologien werden die Websuche unterstützen bei der Interpretation der gefundenen Information und der Kommunikation mit anderen Agenten.
- Logik wird für die Bearbeitung der gefundenen Information und für Schlussfolgerungen benutzt.

Architektur des Semantic Webs

Sprachen für das Semantic Web

- XML - oberflächige Syntax für strukturierte Dokumente. Die Sprache kann aber keine Iinformation über die Bedeutung dieser Dokumente geben
- XML Schema - beschreibt syntaktische Regeln für XML tags.
- RDF - ist ein Datenmodell für Objekte und Beziehungen zwischen Objekten
- RDFS - beschreibt Merkmale und Beziehungen zwischen RDF-Objekte
- OWL ist eine vollständige Sprache für Ontologie- darstellung

„Layer-cake“ Architektur (nach Tim Berners-Lee)

<table>
<tr><td colspan="3">Trust</td></tr>
<tr><td colspan="3">Proof</td></tr>
<tr><td colspan="3">Logic</td></tr>
<tr><td rowspan="5">XML Familie</td><td colspan="2">Ontologies & Ontology Interchange lang.</td></tr>
<tr><td colspan="2">RDF Schemas</td></tr>
<tr><td colspan="2">RDF, Topic Maps, ännliche Technologien</td></tr>
<tr><td colspan="2">XML-Schemas, Namespaces</td></tr>
<tr><td colspan="2">XML-Dokumente</td></tr>
<tr><td colspan="2">URI</td><td>Unicode</td></tr>
</table>

Kürrze Überblick über Mark-up Sprachen

Was ist SGML ?
(Standard Generalized Markup Language)

Internationaler Standard, der die Regeln beschreibt, mit denen man Strukturen eines Dokuments in diesem selbst beschreiben kann.

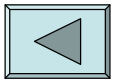
Legt keine spezifische Dokumentstruktur fest, sondern	definiert Regeln und die Syntax zum Aufbau strukturierter Dokumente
SGML-Dokumente sind nicht formatiert oder layoutiert	Die Darstellung der SGML-Dokumente ist Aufgabe eines Browsers
ist kein Format, sondern	Basiert nur auf 7-Bit-ASCII
ist nicht neu, sondern	Wurde 1970 entwickelt für Aufbereitung und Austausch von Rechtstexten

Vorteile von SGML

- Etablierter Standard zur Strukturierung von Dokumenten
- SGML-Dokumente
  - sind plattformneutral
  - besitzen ein einheitliches Erscheinungsbild
  - besitzen eine “Checkliste” für den Redakteur
  - leichter recherchierbar als spezifisch formatierte Texte
  - sind medienneutral aufgebaut
  - benötigen weniger Speicherbedarf als konventionelle Texte
- SGML-Bestandteile sind wiederverwandbar.

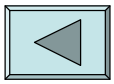
SGML-Dokumente sind plattformneutral

- Da es sich bei SGML-Dokumenten um reine ASCII-Texte mit Strukturauszeichnungen handelt, sind SGML-Dokumente weitestgehend plattform- und softwareunabhängig.
- Neben einer beliebigen Portierbarkeit wird somit die Langlebigkeit von Dokumenten unterstützt, da SGML eben nicht auf einem herstellerspezifischen Speicherformat, speziellen internen Formatierungen oder gar speziellen Hardwarevoraussetzungen basiert.



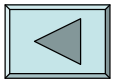
SGML-Dokumente besitzen ein einheitliches Erscheinungsbild

- Durch “formatierungsunabhängige” Struktur-beschreibungen für Dokumente ist ein einheitlicher und konsistenter Aufbau der Dokumente gewährleistet. Prüfprogramme (Parser) gewährleisten die Vollständigkeit und syntaktische Korrektheit der (Instanz-) Dokumente.



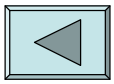
SGML-Dokumente besitzen eine “Checkliste” für den Redakteur

- SGML-fähige Redaktionssysteme, wie z.B. FrameMaker+SGML, unterstützen den Redakteur bei der Erstellung von Dokumenten durch einen kontextsensitiven Elementkatalog



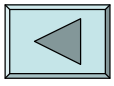
SGML-Dokumente sind leichter recherchierbar

- Gut strukturierte Dokumente ermöglichen eine viel genauere Recherchemöglichkeit von Informationen
- z.B. Eine Volltextrecherche nach dem Begriff *“Werkzeug”* ist unpräziser als die Abfrage *“alle Kapitel die in der Kapitelüberschrift den Begriff “Werkzeug” enthalten”.* Das ist eine Information, die man über das Layout allein nicht erhält.



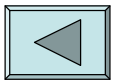
SGML-Bestandteile sind wiederverwendbar

- Da SGML-Dokumente modular aufgebaut sind, lassen sich die Komponenten eines SGML-Dokumentes einzeln ablegen, z.B. In einer SGML-Datenbank, und in unterschiedlichen Kontexten zur Erreichung von Einheitlichkeit gezielt wiederverwerten.
- Beispiele: Definitionen in einem Lehrbuch, Ergebnisse von Bundesligaspielen, Loseblattsammlung mit Verordnungen



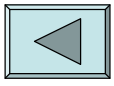
SGML-Dokumente sind trägerneutral aufgebaut

- SGML eignet sich hervorragend für eine träger- (medien-) neutrale Informationsaufbereitung.
- Ein Browser zeigt ein SGML-Dokument an, daraus kann man z.B. drucken
- SGML-Dokumente können z.B. zusammen mit einem geeigneten Browser auf eine CD gebracht und verteilt werden.
- SGML-Dokumente können ohne großen Aufwand nach HTML konvertiert werden und damit Internet-fähig sein.



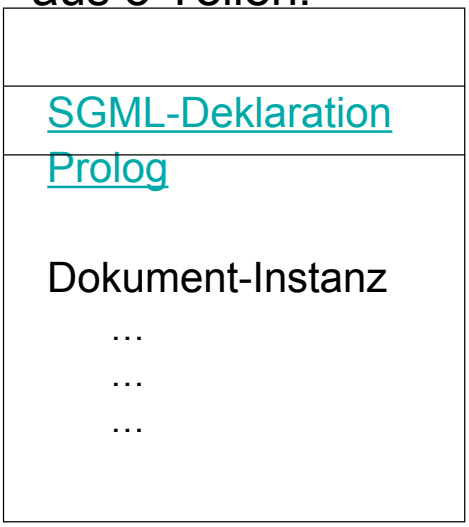
SGML-Dokumente benötigen weniger Speicherbedarf

- SGML-Dokumente sind nur ASCII-Dokumente deshalb ist der Speicherbedarf um ein Vielfaches geringer als bei herkömmlichen Dokumenten aus einer Textverarbeitung.



Aufbau von SGML-Dokumenten

- Ein vollständiges SGML-Dokument besteht aus 3 Teilen:



Der Prolog einer Markup-Sprache hat nichts mit der Programmiersprache „Prolog“ zu tun!

SGML-Deklaration

- Beschreibt formal die für ein bestimmtes Dokument verwendeten Teile des gesamten SGML-Standards,
- beinhaltet Informationen für die Weiterverarbeitung von SGML-Dokumenten (z.B. durch externe Dateien),
- wird vom SGML-System gelesen und interpretiert,
- definiert:
  - den Dokumentzeichensatz, d.h. die als Markierung zu interpretierenden Zeichen,
  - die zulässige Schachtelungstiefe von Strukturen usw.,
- Ist sehr schwer für SGML-Neulinge zu durchschauen,
- viele SGML-Systeme (z.B. FrameMaker+SGML) greifen deshalb auf eine Standard-SGML-Deklaration zurück.

Der Prolog und die DTD (Dokument Type Definition)

- Der Prolog enthält die Strukturregeln (DTD) für ein SGML-Dokument
- DTD = die Definition der Struktur und der Strukturelemente für eine Klasse von SGML-Dokumenten.

DTD Datei-Beispiel: Datei **anthologie.dtd:**

```
<!ELEMENT anthologie      - - (gedicht+)>
<!ELEMENT gedicht         - - (titel?, strophe+)>
<!ELEMENT titel           - O (#PCDATA)>
<!ELEMENT strophe         - O (reihe+)>
<!ELEMENT reihe           O O (#PCDATA) >
```

Der Prolog zitiert dann die Datei **anthologie.dtd** :

Prolog Beispiel:

```
<DOCTYPE Anthologie SYSTEM "c:\...\anthologie.dtd">
```


Eine Dokument-Instanz

```
<!ELEMENT anthologie      - - (gedicht+)>
<!ELEMENT gedicht         - - (titel?, strophe+)>
<!ELEMENT titel           - O (#PCDATA)>
<!ELEMENT strophe         - O (reihe+)>
<!ELEMENT reihe           - O (#PCDATA) >
```

```
<anthologie>
<gedicht>
<titel>An die Muse
<strophe>
<reihe> Was ich ohne dich wäre, ich weiß es nicht - aber mir grauet,
<reihe> Seh ich, was ohne dich Hundert' und Tausende sind.
</strophe>
</gedicht>
<!-- ...andere Gedichte ... -->
</anthologie>
```

Die Syntaxkorrektheit wird von einem Parser überprüft

SGML-Konstrukte

- Erscheinen im DTD-Teil.

Elemente: legen die Bestandteile des Dokumentinhaltes fest
<!ELEMENT gedicht - - (titel?, strophe+)>

Attribute: geben zusätzliche Informationen zu einem Element

<!ATTLIST gedicht sprache (deutsch | englisch) deutsch>

In der Dokumentinstanz: <gedicht sprache=englisch>

Text </gedicht>

Entitäten: abstrakte Bezeichnung für Daten

<!ENTITY uuml “ü”>

In der Dokumentinstanz: <reihe>.. ist für... </reihe>

SGML-Tools

- SGML-Parser: überprüfen das SGML-Dokument hinsichtlich:
  - der Gültigkeit der DTD im Sinne von SGML
  - der Konformität der Dokumentinstanz bezüglich DTD
- SGML-Browser: Anzeigesysteme für SGML-Dokumente
- SGML-Editoren:
  - native SGML-Editoren (z.B. look and feel einer Datenbankoberfläche)
  - WYSIWYG - Editoren

SGML-basierte Anwendungen

- MARTIF ist ein SGML-basiertes Austauschformat für terminologische Daten
  - fachsprachliche Kommunikation braucht korrekte Terminologie.
  - Abhilfe für die Probleme
    - traditionelle Medien (Fachwörterbücher, Glossare usw.) wurden durch Entwicklungen im Bereich der elektronischen Datenverarbeitung stark verdrängt
    - Terminologiedatenbanken wurden von jeder Nutzergruppe anders definiert.
  - 1997 wurde das Terminologie-Austauschformat MARTIF (MAchine-Readable Terminology Interchange Format)definiert
  - spezifiziert die DTD des SGML-Dokuments mit den entsprechenden Tags für die Strukturierung der Daten

Martif-Beispiel

```
<martif>
<martifHeader>
.....
</martifHeader>
<text>
<body>
<termEntry>
<descripGrp>
<descrip type='subjectFieldLevel1'>appearence of material</descrip>
<ntig lang=de>
<termGrp>
<term>Opazität</term>
<termNote type='partOfSpeech>n</termNote>
<termnote type='grammaticalGender'>f</termnote>
</termGrp>
<descripGrp>
<descrip type='definition'> Maß für Lichtundurchlässigkeit
</descrip></descripGrp>
</ntig>
.....
```

SGML-basierte Anwendungsklassen

- SGML-basierte bibliographische Datenbank für Nachschlagewerke
- SGML-basierte Publikationprozesse im Verlag
- Semantisches Markup zur Inhaltserschließung von Agenturmeldungen
- Computerunterstützte Textanalyse (Textannotation)

SGML und das Web

- SGML ist sehr gut geeignet für “large-scale document management business”.
- Nicht so gut geeignet für Web-Publikationen denn:
  - Komplexe Software ist erforderlich, denn SGML ist sehr kompliziert und erfordert einen entsprechend komplizierten Parser, der für On-line Arbeiten im Netz (z.B. Browsen) zu langsam sein würde,
  - Nach dem SGML-Standard wäre für jedes Dokument unbedingt eine DTD nötig,
  - die DTDs haben eine sehr komplexe Struktur und sind nicht ohne weiteres wiederbenutzbar oder änderbar,
  - zwei Dokumente mit zwei unterschiedlichen DTDs können nicht gemischt werden.

XML versus SGML

- XML ist ein “application profile” von SGML: ein SGML-System kann XML-Dokumente lesen
- XML ist eine beschränkte Teilmenge von SGML
- XML hat keine eigene Deklaration
- XML kann SGML nicht ersetzen. SGML ist eine bessere und abstraktere Lösung für die Erzeugung und Entwicklung von komplexen Dokumenten und Datenbanken.

Was ist XML ?
(Extensible Markup Language)

- XML ist eine eingeschränkte Markup-Sprache für Dokumente, die strukturierte Information enthalten
- strukturierte Information ist z.B.:
  - Inhalt (Wörter, Bilder, usw.)
  - Hinweis über Inhaltbedeutung (Kapiteltitel, Überschriften, usw.)
- Die XML-Spezifikation definiert einen Standard für vereinfachte Markupentwicklung und ohne eine komplizierte Deklaration.
- Dokumente können sein:
  - Konventionelles Standarddokument
  - Vektor-Graphik
  - e-commerce-Transaktionen
  - Mathemetische Formeln
  - meta-Daten, usw.

XML Tools

- Editor: ein normaler ASCII-Editor
- Dokumente können mit Web-Browsern angesehen werden (z.B. Netscape Navigator ab Version 5, Internet Explorer ab Version 4.5)
- HTML-Tags können innerhalb von XML-Dokumenten benutzt werden.
- XML enthält keinen vordefinierten Tag für Hyperlinks. (dafür muss XML Linking Language - XLS - benutzt werden)
- XML-Dokumente können in HTML-Dateien eingelesen und z.B. mit JavaScript ausgewertet werden.

XML Beispiel

- Syntax ist ähnlich wie SGML.

```
<?xml version”1.0”?>                                          } XML-Prolog
<?xml-stylesheet href=“style.css” type=“text/css”?>
<!DOCTYPE Dokument [                                          }
<!ELEMENT Dokument (Titel, Abstrakt?, Kapitel+, Zusammenfassung, Bibliographie)>
        <!ELEMENT Titel (#PCDATA)>
        <!ELEMENT Abstrakt (#PCDATA)>
        <!ELEMENT Kapitel (Titel, #PCDATA)>                   } DTD
        <!ELEMENT Zusammenfassung (#PCDATA)>
        <!ELEMENT Bibliographie (Reihe+)>
        <!ELEMENT  Reihe (#PCDATA)>
]>                                                            }
<Dokument>                                                    }
<Titel> XML Einf&uuml;hrung</Titel>                           } Dokument-Instanz
<Abstrakt> .....Text .... </Abstrakt>
....
</Dokument>                                                   }
```

Die Formatierung wird danach durch eine CSS - Datei (Cascading Style Sheets) style.css erzeugt.

XML- Erweiterungen

- **LT-XML**
  - Entwickelt an der Universität Edinburgh (Language Technology Group)
  - XML parser + flexible API + Toolsammlung für XML marked-up Dokumenten-Verarbeitung

```
<p id='pl'>
<s id='sl'>
<w pos='prep'>In</w>
<w pos='art'>the</w>
<w pos='n'>beginning</w>
<w pos='v'>was</w>
<w pos='art'>the</w>
<w pos='n'>word</w><c>.</c>
</s>
</p>
```

Beispiel.xml

%textonly -s ' '<sample.xml

In the begining was the world.

%sgcount <sample.xml

p 1
s 1
w 6
c 1

Ergebnisse der Auswertung

%sggrep -q '.*/w[pos="n"]' sample.xml

```
<w pos='n'>beginning</w>
<w pos='n'>word</w>
```


XML vs. HTML

HTML wurde mit SGML definiert

↓

HTML entspricht nicht den XML-Standards

↓

XML-Kode kann nicht in HTML-Dokumente eingebaut werden

MathML
LT-XML
SMIL

↓

XHTML

Was ist XHTML ?

- Redefinition von HTML tags entspechend der XML-Syntax
- die Dokumentstruktur bleibt dieselbe

```
<html>
   <head>
    <!--   ... Head-Inhalt ....-->
   <title> Dokumenttitel </title>
   <!-- ...  Head-Inhalt ... -->
 </head>
<body>
    <!-- ... Body-Inhalt ... -->
</body>
</html>
```

- Einige Tags haben in XHTML **NICHT** dieselben Namen wie in HTML

XHTML vs. HTML

- Alle Tag- und Attribut-Namen müssen in Kleinbuchstaben geschrieben werden (weil XML kleine und grosse Buchstaben unterscheidet)
- Es gibt keine optionalen Ende-tags. Alle Tags müssen ein Paar sein (also auch z.B. ...** **)
- Alle leeren Tags enthalten wie in XML ein Leeres-Element-Tag)
  <hr **/**>
- es gibt nur ein einziges `head`- und ein einziges `body`- Element. Stattdessen kann man nur ein einziges `frameset` -Element einfügen
- jedes `head`-Element darf nur ein einziges `title` -Element (Tag) enthalten

XHTML vs. HTML (Beispiel)

```
<html>
<head>
<title> Vorlesung CP
    Content</title>
</head>
<body>
<h1> Vorlesung CP </h1>
<hr / >
<h2>Inhalt</h2>
<ul>
   <li>01 Intro </li>
   <li> 02 Theorie </li>
</ul>
</body>
</html>
```

```
<HTML>

<body>
<h1> Vorlesung CP </h1>
<hr>
<h2>Inhalt</h2>
<ul>
   <li>01 Intro
  <li> 02 Theorie
</ul>
</body>
</html>
```

Viele HTML-Editoren ergänzen HTML zu XHTML

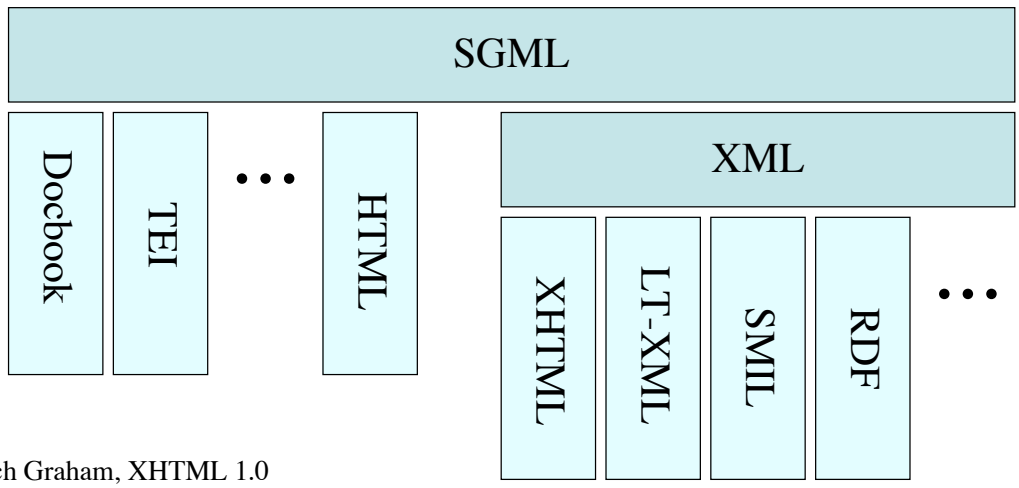
Strictly Conforming XHTML

- Entspricht einem strikten XML-Formalismus:
  - Spezifiziert, dass das Dokument vollständig XML-formatiert ist:
    <?xml version="1.0" charset="iso-8859-1" ?>
  - benennt eine DTD
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict// EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd" >
  - das <html> Element muß ein "xmnls" Attribut enthalten, um zu spezifizieren wo die Elementnamen definiert sind:
    <html xmnls="http://www.w3.org/TR/xhtml1">
    ...
    </html>

Strictly Conforming XHTML - Beispiel

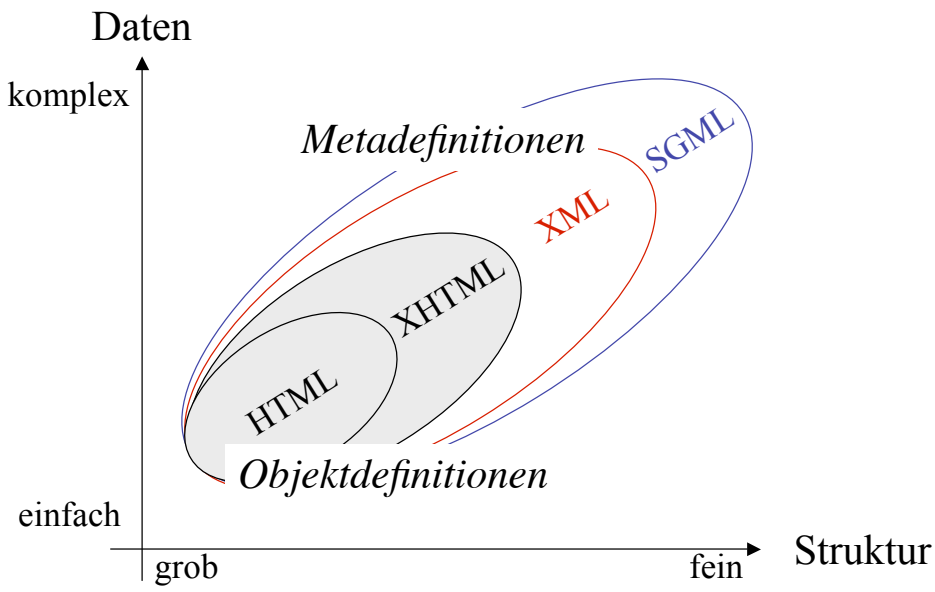
```
<?xml version="1.0" charset="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//
    EN"
 "http://www.w3.org/TR/xhtml1/DTD/strict.dtd" >
<html xmnls="http://www.w3.org/TR/xhtml1">
<head>
<title> Vorlesung CP Content</title>
</head>
<body>
.....
</body>
</html>
```


Derivate von SGML und XML



Nach Graham, XHTML 1.0

SGML, XML, XHTML und HTML



Bestandteile von XML

- Vorspann:
  - Version: z.B. <?xml version="1.0">
  - DTD: Verweis auf Grammatik: <!DOCTYPE…>
- Elemente (Tags): z.B. <offer>,<store>,<book> usw.
  - Änlich wie in HTML aber selbstdefiniert
- Attribute: z.B. für >price>: type="retail".
  - Änlich wie in HTML aber selbstdefiniert
- Entitäten
  - Zur Wiederverwendung und modularisierung von Texten

Elemente

- Elemente sind die grundlegenden Komponenten eines XML-Dokuments
- Sie werden durch beliebige Namen eingeschlossen in **<** und **>** repräsentiert.
- D.H. zur Definition eines neues Elements definiert man einen neuen Tag sowie den zugehörigen Abschlusstag
- Elemente können innerhalb von anderen Elementen definiert werden (Hierarchien)
- Namenskonventionen wie bei der Programmierung

Beispiel

```
<?xml version="1.0">
<kontakt>
      <name>
             <vorname>Peter </vorname>
            <nachname>Mustermann</nachname>
     </name>
    <email>peter@firma.de</email>
   <telefon> 0236589 </telefon>
</kontakt>
```

- Wurzelelement: `kontakt`
- Elemente: `name, vorname, email, telefon`
- Drei Hierarchieebenen

Attribute

- Ein Element kann nicht nur Unter-elemente besitzen, sondern auch durch Attribute erweitert werden.
- Ein Attribut wird samt seinem Wert (immer in anführungszeichen) dem Start-Tag eines Elements hinzugefügt.
- Beispiel: ein Kapitel eines Buches hat einen Title und eine Nummer:

```
<chapter title="Introduction" number="4">
….
</chapter>
```


Wann Element, wann Attribut?

- XML gestattet die Datenformatierung auf viele Arten; u.a. ist es praktisch immer möglcih anstelle von Attributen auch Elemente zu verwenden und umgekehrt.
- Einiege Tipps:
  - Wenn die information selbst wieder komplex ist verwendet man Elemente
  - Modelllierung geordneter Information: Elemente
  - Leserlicher ist die Elementvariante, sie nimmt deshalb auch mehr Platz ein.
  - Ist die modellierte Information ein Bestandteil des übergeordneten Elements, verwendet man ebenfalls ein element. Ist sie eine Eigenschaft (wie etwa eine Farbe), dann verwendet man Attribute.

Validierung

- Die strengen Syntaxkonventionen erleichtern die Programmierung von XML Applikationen bereits erheblich (wohlgeformte Dokumente)
- Validierung erlaubt es, die möglichen Baumstrukturen einzuschränken (valide Dokumenten)
- Validierung findet immer gegenüber einer Grammatik statt, d.h., ein valides Dokument ist:
  - Wohlgeformt und
  - Konform zu einer vorgegebenen Grammatik
- Ein grosser Teil der Fehlerabfragen kann so von der applikation an einen Parse übertragen werden

Definition von Grammatiken

- Wenn zwei Anwendungen dieselbe Grammatik verwenden, haben sie dasselbe Vertändnis von Daten (wenigstens die Hierarchie)
- Es werden heute zwei verschiedene ansätze verwendet, um eine Grammatik zu beschreiben:
  - Document Type Definition (DTD)
  - XML Schema

XML Schema als Alternative zur DTD

- Die DTD-Syntax ist nicht XML-konform (sondern SGML)
- Keine Datentypen, nur Strings
- Begrentzte Erweiterbarkeit (vererbung)

- XML Schema behebt diese Probleme
  - Das Konzept bleibt dasselbe
  - Verschiedene Datentypen
  - Geschrieben in XML
- Vorgehen: schreibe ein Schema in XML, referenziere es von der XML-Datei aus.

Vorteile von XML-Schema

- Jedes XML Schema ist selbst ein XML-Dokument (im Gegensatz zu DTD keine spezielle Syntax mit speziellen Verarbeitungswerkzeugen erforderlich)
- Auch komplexe Integritätsbedingungen formulierbar
- XML Schema enthält vordefinierte und eigendefinierbare Datentypen, wodurch Typprüfung möglich wird
- Bei Datentypen werden Vererbung und substitution unterstützt.
- Benennungskonflikte können durch Verwendung von XML-Namensräumen vermieden werden.

Beispiel für XML-Schema -1-

```
<xsd:schema xmlns:xsd=http://www.w3.org/2002/XMLSchema”>
<element name=“paper” type=“papertype” />
<xsd:complextype name=“papertype”>
  <xsd:sequence>
      <xsd:element name=“autor” type=“xsd:string”/>
      <xsd:element name=“titel” type=“xsd:string”/>
      <xsd:element name=“datum” type=“xsd:gYearMonth”/>
      <xsd:element name=“link” type=“xsd:anyURI min Occurs=“0”/>
  </xsd:sequence>
 <xsd:attribute name=“typ” type=“art”/>
</xsd:complextype>
```


Beispiel für XML-Schema -2-

```
<xsd: simpleType name=“art”>
     <xsd: restriction base=“xsd:string”>
          <xsd:enumeration value=“proc”/>
          <xsd:enumeration value=“report”/>
          <xsd:enumeration value=“journal”/>
           <!-- usw. -->
     </xsd:restriction>
</xsd:simpleType>
```

Parser

- Um ein XML-Dokument in einer Anwendung zu verwenden, muss man es einlesen und in interne Datenstrukturen der verwendetenProgrammiersprache umwandeln
- Es werden heute wesentlichen zwei Modelle für das Parsen von XML verwendet:
  - Simple API für XML (SAX)
  - Document Object Model (DOM)
- JavaXML-library bietet Lösungen für beides an.

Wie verarbeiten wir die Daten?

- Möglichkeit 1: schreibe eine anwendungfür den verwendeten Parser.
  - Z.B: lese Dokument als DOM-Baum ein, bearbeite dann den Baum
- Möglichkeit 2: bearbeite das Dokument mittel vorgegebener Regeln
- Hierzu verwendet man sog. Stylesheet-Transformationen (XSL)

Zusammenfassung

- XML an sich ist eine sehr grundlegende Technologie
- Die Stärken liegen in:
  - Der grossen Akzeptanz
  - Der enormen Menge an frei erhältlicher Software
  - Der Vielzahl der Anwendungsbereiche

Themen in Proseminar

Termine und Themen

- 24.04 Einführung
- 08.05 Document-Typen Mark-up Sprachen in WWW: XHTML, CSS
- 15.05 Metadaten in WWW: XML und XML-tools
- 22.05 Semantische Repräsentation der Daten in WWW (RDF)
- 29.05 Ontologische Repräsentation von Daten (OWL)
- 05.06 Pfingsten
- 12.6 Klassische Verfahren des IR und Ihre Anwendung bei Suchmaschinen
- 19.06 Natürlichsprachliche Schnittstellen
- 26.06 Informatikstatistische Verfahren
- 03.07 Ranking
- 10.07 Zusammenfassung

Scheinkriterien

- Anwesenheit (maximal 2 motivierte Abwesenheiten)
- Vortrag
- HTML Version des Referats, folgend ein gegeben CSS.