

Rückblick

Ausblick

Hinweise für Klausur

Heutige Vorlesung

Rückblick

- Kurze Wiederholung von Themen, die bei Prüfungen Probleme bereiteten.

Ausblick

- XSL-FO: Ergänzung zu XSLT
- Relax NG: Alternative zu XML-Schemata?
- Semantic Web: Alternative zu XML?
- Netzwerkeffekte: Wie setzen sich Standards durch?

Klausur

Rückblick

Rückblick

XML

- Semantik, Daten als XML speichern?

XML-Schema

- Validierung auf mehreren Ebenen, Typsubstitution

XSLT

- Prozessor, leeres Stylesheet, Mächtigkeit

Web-Dienste

- Aufbau WSDL, Dynamisches Einbinden, RPCs vs. Messaging

Anwendungen

XML

Semantik in XML

- Bedeutung von `<p>Text</p>`

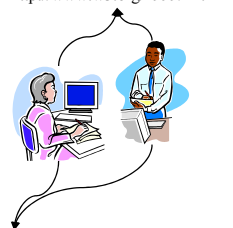
- in HTML: fest
- in XML: offen

- XML-Schema: Inhalt des Elementes ein Datentyp (und damit eine gewisse Bedeutung) zuordnen

- Namensraum: Element selbst Bedeutung geben

`<p xmlns="http://www.w3.org/1999/xhtml">Text</p>`

HTML auf bestimmte Semantik festgelegt, XML nicht.



Daten als XML speichern?

```

<Employee>
  <EmployeeNo>4</EmployeeNo>
  <Name>
    <First>Mark</First>
    <Last>Whitehorn</Last>
  </Name>
  <City>
    <Name>New York City</Name>
    <CityId>NYC</CityId>
  </City>
  <Type>Sales Person</Type>
  <Orders>
    <Order>
      <OrderNo>121</OrderNo>
      <OrderItems>...</OrderItems>
      <CustomerNo>999</CustomerNo>
    </Order>
  </Orders>
</Employee>
    
```

- als Austauschformat OK
- als Speicherformat aber problematisch:
 - Änderungsanomalie
 - Löschanomalie

Daten als XML speichern?

Daten mit vielen funktionalen Abhängigkeiten

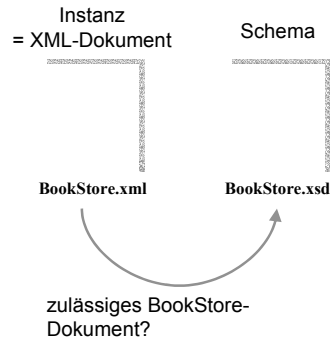
- besser professionelle Speicherformate wählen (z.B. relationale Datenbanken).

Text-Dokumente ohne funktionale Abhängigkeiten

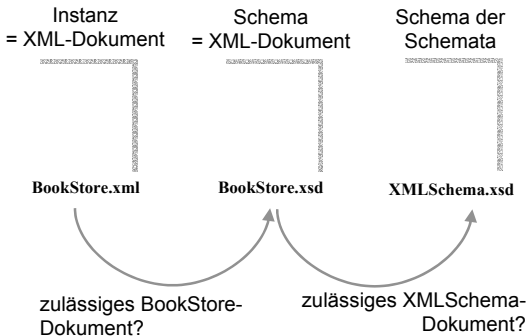
- XML kann als Speicherformat benutzt werden.

XML-Schemata

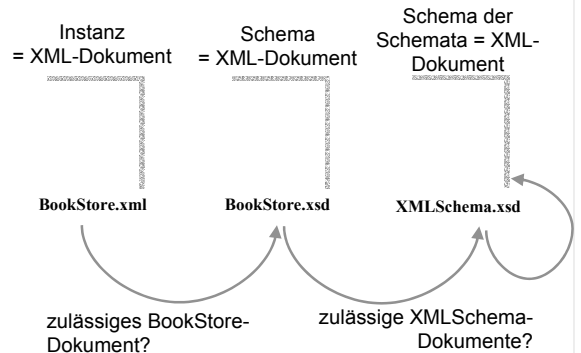
Validierung



Validierung auf mehrere Ebenen



Validierung auf mehrere Ebenen



Abgeleiteter Datentyp

```
<xsd:complexType name="NameType">
  <xsd:sequence>
    <xsd:element name="first" type="xsd:string"/>
    <xsd:element name="middle" type="xsd:string"/>
    <xsd:element name="last" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="title" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="ExtendedNameType">
  <xsd:complexContent>
    <xsd:extension base="target:NameType">
      <xsd:attribute name="gender" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Datentyp t

Datentyp t' mit
zusätzlichem
Attribut gender

Beispiel einer Typsubstitution

```
<xsd:element name="name" type="NameType">
```

Schema

Instanz

```
<name title="Mr.">
  <first>...</first>
  <middle>...</middle>
  <last>...</last>
</name>
```

oder

Instanz

```
<name title="Mr." gender="female" xsi:type="ExtendedNameType">
  <first>...</first>
  <middle>...</middle>
  <last>...</last>
</name>
```

Typsubstitution

- für Einschränkungen (echte Teilmengen) unproblematisch
- für Erweiterungen evtl. problematisch
- block="extension"**: Typsubstitution wird für Erweiterungen unterdrückt, z.B.:

```
<xsd:element name="name" type="NameType"
  block="extension">
```

XSLT

Wie arbeitet ein XSLT-Prozessor?

Stylesheet

```
<xsl:template match="A">
  <xsl:value-of select="@id"/>
</xsl:template>
<xsl:template match="B">
  <xsl:value-of select="@id"/>
</xsl:template>
<xsl:template match="C">
  <xsl:value-of select="@id"/>
</xsl:template>
<xsl:template match="D">
  <xsl:value-of select="@id"/>
</xsl:template>
```

```
<source>
  <A id="a1">
    <B id="b1"/>
    <B id="b2"/>
  </A>
  <A id="a2">
    <B id="b3"/>
    <B id="b4"/>
    <C id="c1">
      <D id="d1"/>
    </C>
    <B id="b5">
      <C id="c2"/>
    </B>
  </A>
</source>
```

kein Template
anwendbar

Template "A"
wird
angewandt

a1
a2

Template "B"
wäre anwendbar,
es werden aber
keine Templates
aufgerufen!

Leeres Stylesheet

- Bei Stylesheet *ohne* Templates sind nur die beiden vordefinierten Templates aktiv:

```
<xsl:template match="*" />
<xsl:apply-templates/>
</xsl:template>
<xsl:template match="text()|@">
  <xsl:value-of select="."/>
</xsl:template>
```

- Gesamtes Ursprungsdokument wird traversiert, dabei werden Text-Inhalte und Attribut-Werte extrahiert:

```
<name>
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

John
Fitzgerald Johansen
Doe

Mächtigkeit von XSLT

- Variablen machen Stylesheets zu einem mächtigen Termersetzungssystem mit unbeschränkten Registern.
- www.unidex.com/turing definiert universelle Turingmaschine in XSLT.
- Damit wird der IE zum vollwertigen Computer!
- Stylesheets tatsächlich berechnungsvollständig und damit eine vollwertige Programmiersprache (Kepser 2002).
- **Problem:** Terminierung von Stylesheets kann *nicht* garantiert werden.

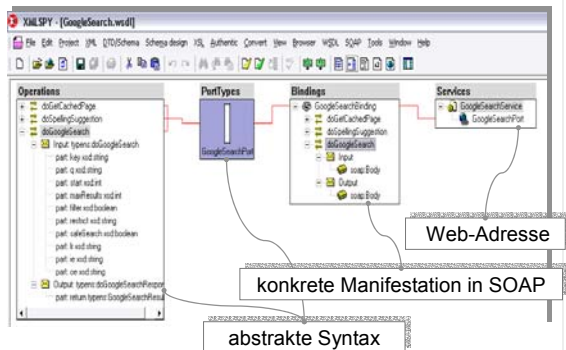
Principle of Least Power

When I designed HTML for the Web, I chose to avoid giving it more power than it absolutely needed - a "principle of least power," which I have stuck ever since. I could have used a language like Donald Knuth's "TeX," which though it looks like a markup language is in fact a programming language. It would allow you to express absolutely anything on the page, but would also have allowed Web pages that could crash, or loop forever (Tim Berner-Lees, 1999).

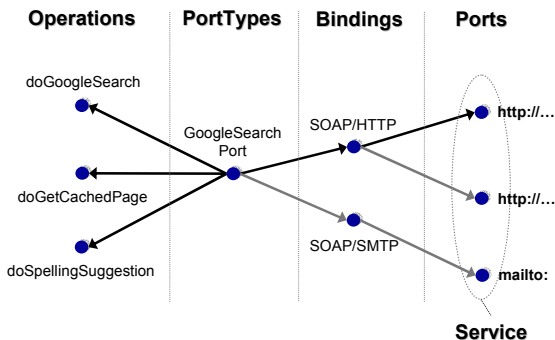
Verletzt XSLT dieses Prinzip?

Web-Dienste

Die WSDL-Beschreibung von Google



Grundstruktur WSDL



RPC vs. Messaging

RPC

→ eng gekoppelte, starre Systeme

- + einfach, abstrahiert von Kommunikation
- nur Eins-zu-Eins-Kommunikation
- Client und Server müssen präsent sein.
- nicht skalierbar

Messaging

→ lose gekoppelte, flexible Systeme

- abstrahiert nicht von Kommunikation
- + erlaubt auch One-to-Many-Kommunikation
- + Weder Sender noch Empfänger müssen präsent sein.
- + erlauben Priorisierung und Lastverteilung, dadurch skalierbar

Einbindung eines Web-Dienstes

zur Laufzeit

→ Vision

- vollautomatisches Einbinden prinzipiell mit WSDL möglich, wird aber die *Ausnahme* bleiben.

zur Entwicklungszeit

→ Realität

- kein vollautomatisches Einbinden, WSDL *erleichtert* aber das Einbinden ganz erheblich:
- Web-Dienst finden: nicht automatisiert
- Web-Dienst aufrufen: teilweise automatisiert

Aufruf eines Web-Dienstes

- gegeben: WSDL-Beschreibung
- Aus der WSDL-Beschreibung kann *automatisch* eine Schnittstelle zur Anwendung (Stubs) generiert werden: z.B. eine Methode setParameterQ einer Klasse DoGoogleSearch
- Abbildung zwischen Stubs und Anwendungslogik normalerweise *nicht* automatisierbar:
Wie und wo SetParameterQ im Anwendungsprogramm aufrufen?
 - nur teilweise automatisierbar
 - Ausnahme: Operationen und Parameter sind standardisiert

Anwendungen

Kategorien von XML-Anwendungen

XML als Meta-Sprache

- Mit XML-Schema/DTD wird Auszeichnungssprache für spezielle Anwendung definiert.

Lösung konkreter Probleme

- XML-Technologien werden zur Lösung konkreter Probleme eingesetzt.

XML als Meta-Sprache

- **XHTML**: Reformulierung von HTML in XML
- **WML**: Wireless Markup Language
Präsentation von Inhalten auf mobilen Endgeräten
- **DocBook**: strukturierte Darstellung von Bücher/Artikel
- **MathML**: Mathematical Markup Language
Standard für mathematische Ausdrücke
- **SVG**: Scalable Vector Graphics
Standard für Vektorgraphiken
- **SMIL**: Synchronized Multimedia Integration Language
Standard für Multi-Media-Anwendungen
- **VoiceXML**: Voice Extensible Markup Language
Standard für interaktive Sprachanwendungen

XML als Meta-Sprache

- **XHTML**: Reformulierung von HTML in XML
- **WML**: Wireless Markup Language
Präsentation von Inhalten auf mobilen Endgeräten
- **DocBook**: strukturierte Darstellung von Bücher/Artikel
- **MathML**: Mathematical Markup Language
Standard für mathematische Ausdrücke
- **SVG**: Scalable Vector Graphics
Standard für \forall **unterschiedliche Anwendungen,**
- **SMIL**: Synchronized
Standard für \forall **aber einheitliche Syntax**
- **VoiceXML**: Voice Extensi
Standard für \forall **nur ein Parser nötig**
echte Erfolgsgeschichte!

XML als Lösung konkreter Probleme

Trennung von Inhalt und Präsentation

- Inhalt: XML
- Präsentation: HTML, WML, ASCII, ...
- Transformation: XSLT, Perl, ...

Semantic Web

- komplexe Anfragen an das Web stellen

Web-Dienste

- verteilte Systeme im Web realisieren

Ausblick

Was gibt es noch?

XSL-FO Formatting Objects

- Ergänzung zu XSLT

Relax NG

- Alternative zu XML-Schemata?

Semantic Web

- Alternative zu XML?

Netzwerkeffekte

- Wie setzen sich Standards durch?

XSL-FO

XSL Formatting Objects (XSL-FO)

XSLT

- erlaubt Transformation von XML → HTML
- ungeeignet für druckfähige Formattierungen (PDF, RTF)

XSL-FO

- erlaubt XML-Dokumente mit druckfähigem Layout zu versehen
- Transformation XML → PDF oder RTF möglich
- basiert auf auf Cascading Style Sheets (CSS2)
- W3C-Standard von 2001

→ <http://www.w3.org/TR/xsl/slice3.html#fo-jc-intro>

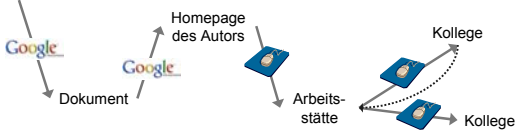
XSL = XSLT + XSL-FO

Was leistet XSL-FO?

The screenshot displays two panels of XSL-FO output. The top panel, titled 'font-stretch', shows a list of relative keyword values for the following styling: font-stretch to normal, 1. ultra-condensed, 2. extra-condensed, 3. condensed, 4. semi-condensed, 5. semi-expanded, 6. expanded, 7. extra-expanded, and 8. ultra-expanded. The bottom panel, titled 'The relative keyword values:', shows a list of relative keyword values: 1. normal, 2. ultra-condensed, 3. extra-condensed, 4. condensed, 5. semi-condensed, 6. semi-expanded, 7. expanded, 8. extra-expanded, and 9. ultra-expanded. The output is styled with various font weights and sizes, demonstrating the capabilities of XSL-FO.

Semantic Web: Berner-Lees' Vision

- Webinhalte und ihre Vernetzung werden für Maschinen verständlich.
- Komplexe Anfragen können ans Web gestellt werden, die heute nur durch Surfen zu beantworten sind.
- Beispiel:** Mit welchen Kollegen arbeitet der Autor eines bestimmten Dokumentes zur Zeit zusammen?



Erste Schritte zum Semantic Web

1. XML + Namensräume + XML-Schema

- Webseiten ⇔ maschinenverarbeitbare Daten
- Definition eines Vokabulars für Daten

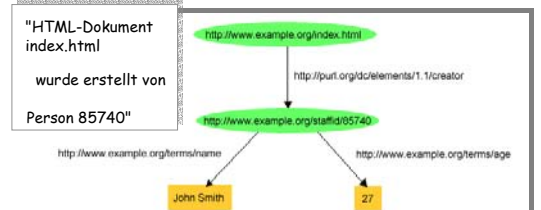
2. RDF + Namensräume + RDF-Schema

- Hyperlinks ⇔ maschinenverarbeitbare Beziehungen
- Definition eines Vokabulars für Beziehungen

RDF

- W3C-Standard von 2004
- verschiedene Versionen:
 - Tripel: kompakt, lesbar
 - RDF/XML: für maschinelle Verarbeitung
- Tripel setzen bel. Webressourcen miteinander in Beziehung, z.B.:
 - <URI-1, ist-Kollege-von, URI-2>
- Triple können an bel. Stelle (URI) abgespeichert sein:
 - URI: <URI-1, Relation, URI-2>

Beispiel



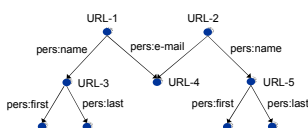
- Relationen auch durch URI identifiziert
- Falls ein Programm diesen Namensraum kennt, kennt es auch die Bedeutung der Beziehung.
- weitergehende Infos: <http://www.w3.org/TR/rdf-primer/>

XML vs. RDF

XML

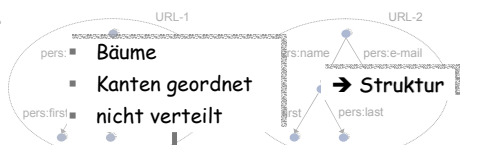


RDF



XML vs. RDF

XML



XML = Spezialfall von RDF

RDF



Semantic Web: Die Realität



XML

- XML-Dokumente werden erstellt und benutzt. ✓

RDF

- Leute erstellen HTML-Seiten und XML-Dokumente, aber kein RDF. Wozu auch?
- kein Migrationspfad: Wie kommt man vom heutigen Web zu einem Semantic Web?

für RDF bisher keine
Killerapplikation

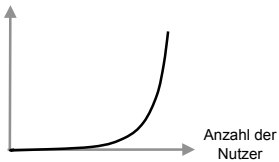
Netzwerkeffekte



Netzwerkeffekte



Nutzen für
einzelnen Nutzer

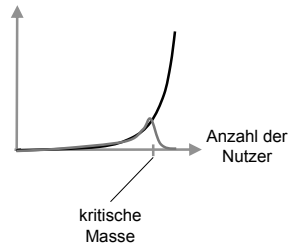


- Beispiele:** Telefon, E-Mail, HTML, XML
- Gibt es nur einen Nutzer, ist Nutzen für diesen Nutzer gleich Null.
- Je mehr Nutzer, desto höher Nutzen für einzelnen Nutzer.

Kritische Masse



Nutzen für
einzelnen Nutzer

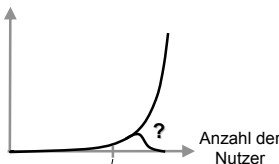


- Unterhalb kritischer Masse: Nutzen sehr gering
- Oberhalb kritischer Masse: Nutzen steigt rapide an
- Bei konkurrierenden Netzwerken: *nicht* vorhersagbar, welches Netzwerk sich durchsetzen wird.

Netzwerkeffekte XML



Nutzen für
einzelnen Nutzer



XML 2004

- XML hat noch *nicht* eine kritische Masse von Nutzern erreicht.
- Aber:** Zum ersten mal in der Geschichte haben sich *alle* großen IT-Unternehmen auf einen Standard geeinigt: XML, SOAP und WSDL

Klausur



Klausur



- **Termin:** 21.7.
- **Zeit:** 12:15-13:45
- **Raum:** SR 005
- **bei Nichterscheinen:** *keinen* Schein
- **einzige Ausnahme:** Vorlage eines ärztlichen Attestes
- **falls nicht bestanden:** mündliche Prüfung Anfang WS 04/05
- **Plagiate:** alle Beteiligten nicht bestanden
- **Hilfsmittel:** keine, außer Papier und Stift
- **mitbringen:** Stift, Ausweis
- **Sprache:** Antworten auf Deutsch oder Englisch

Klausurrelevante Unterlagen

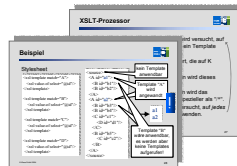


- alle Foliensätze
- Hunter et al., *Beginning XML* (2nd Edition), Wrox Press, 2001: S. 29-147.
- <http://www.w3.org/TR/xmlschema-0/> oder S. 217-288 aus Hunter et al. (2001).
- <http://java.sun.com/webservices/docs/1.2/tutorial/doc/>: Kapitel 1, Unterkapitel The "SAX API" und "The DOM API".
- <http://www.w3.org/TR/soap12-part0/>
- <http://www.w3.org/TR/wsdl>

Klausurthemen



- Fragen/Aufgabenstellungen beziehen sich auf Vorlesungsstoff, insbesondere auf Foliensätze
- Für Verständnis der Foliensätze jedoch Lektüre der genannten Literatur häufig unerlässlich.
- **Beispiel: Wie geht ein XSLT-Prozessor vor?**
 - war Thema einer Vorlesung
 - dazu gibt es mehrere Folien
 - wenn nicht klar, dann gezielt nachlesen



Wie geht es weiter?



- heute 16:15 betreute Rechnerübung
- **Klausur**
 - 21.7. von 12:15-13:45 im SR 005
- **Sprechstunden**
 - für Fragen, die sich bei Klausurvorbereitung ergeben
 - ohne Voranmeldung:
 - Mittwoch den 14.7. von 16:00-18:00
 - Montag 19.7. von 12:00-14:00
 - Dienstag 20.7. von 10:00-12:00
 - jeweils in der Fabbeckstr. 15

Wintersemester 2004/2005



PJ XML-Projekt (2 SWS)

- Entwicklung eines XML-basierten Content-Management-Systems
- Einübung von industrieller Projektarbeit
- Ich: Auftraggeber (= Kunde)
- Sie: Projektleiter, SW-Architekt, Programmierer
- Voraussetzungen
 - HTML
 - XML, XML-Schemata, SAX, DOM, XSLT, SOAP, WSDL
 - Java

Wintersemester 2004/2005



S Geschäftsmodelle und Existenzgründung in der IT-Industrie (2 SWS)

- theoretischer Teil (= Referate)
 - Erfolgreiche Geschäftsmodelle
 - Zentren der Existenzgründung
 - Goldene Regeln der Existenzgründung
 - Businessplan
 - Finanzierung
 - Rechtsformen
 - Preisgestaltung
- praktischer Teil
 - Businessplan entwerfen
 - kleiner Businessplanwettbewerb