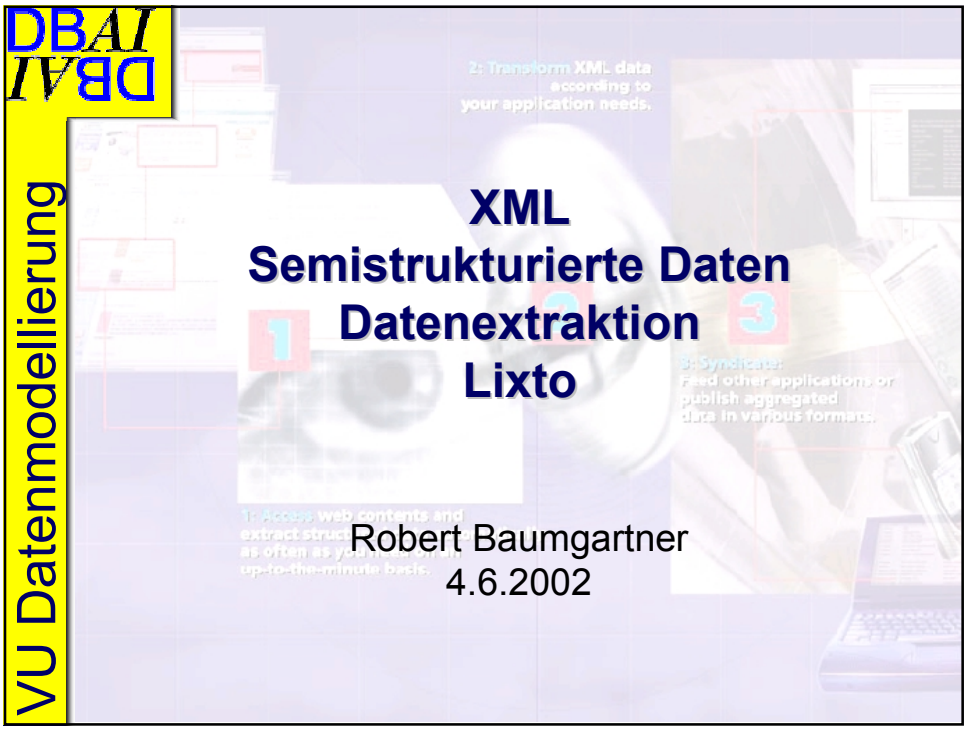
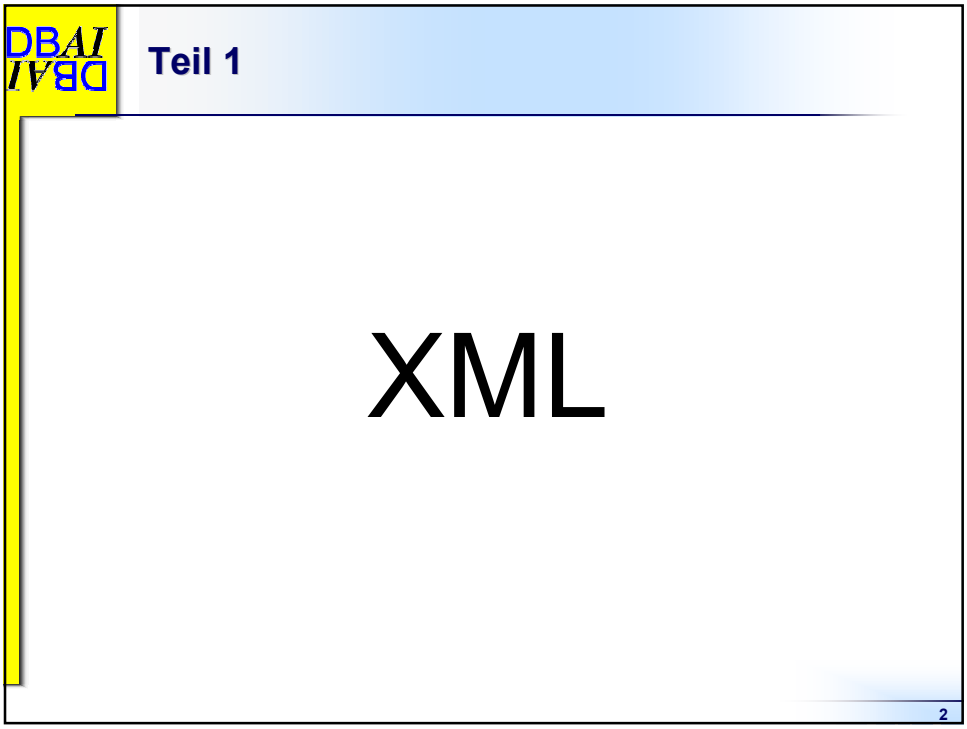

DBAI
VU Datenmodellierung
XML
Semistrukturierte Daten
Datenextraktion
Lixto
Robert Baumgartner
4.6.2002


DBAI
Teil 1
XML
2

Motivation

- Informationsaustauschformate
- Zusammenwachsen der Dokumentwelt und Datenbankwelt
  - Datenbankwelt: semistrukturierte Datenmodelle
  - Dokumentwelt: XML
- Web als große Datenbank
- Datenmengen mit flexibler und unregelmäßiger Datenstruktur

Dokumentwelt

- Intra- und Interdokumentstruktur
- Präsentationsformate wie HTML
  - die nur sehr grobe Struktur aufweisen
  - basierend auf SGML (s.u.)
- globale Infrastruktur für Dokumentaustausch (z.B. Web)
  - Web ist strukturlos, bestenfalls ein großer Graph
- Mischformen zwischen Dokumenten und Datenbanken
  - Sammlungen wie Steuergesetzgebungen, Jura ... (tw. in SGML)
- Management von großen Dokumentsammlungen
- Dies führte zur Notwendigkeit neuer Formate zum Datenaustausch mit Struktur, insbesondere zur Entwicklung von XML.

Datenbankwelt: Stand der Dinge

- Relationale DB, EER zur Strukturbeschreibung
  - modelliert als endliche First Order Logik Struktur
  - In DB Theorie spielt die endliche Modelltheorie (FMT) und deskriptive Komplexitätstheorie eine Rolle
- Datenmodelle und Abfragesprachen
- Trennung logische Sicht vs. physikalische Implementierung
  - logische Sicht: was sind gültige Abfragen,...
  - physik. Implementierung: wie werden Daten gespeichert,...
- externe Sicht: Views wie welcher Benutzer welche Daten wahrnimmt
- Speichertechniken und Techniken für Datenbankkonsistenz/Integrität

5

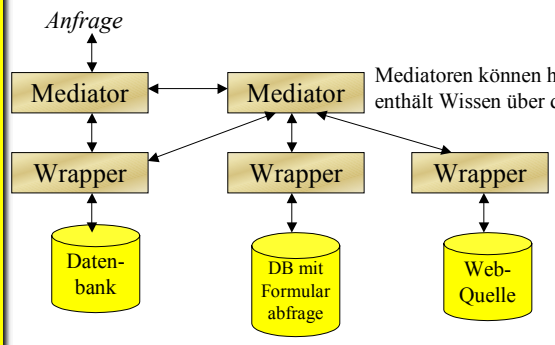
Semistrukturierte Daten

- Die Suche nach Modellen für flexible und unregelmäßige Datenstruktur führte zu Modellen für semistrukturierte Daten
  - wenn kein festes bekanntes explizit gegebenes Schema
  - wenn Datenbanken mit vielen Nullwerten
  - wenn große Datenbankeschemata
  - wenn Daten nicht sehr typisiert (i.e. kann von verschiedenem Typ sein)
- Modellierung als Graph
- Semistrukturierte Daten (SSD) oft als selbstbeschreibend bezeichnet
  - keine eigene Beschreibung der vorgeschriebenen Struktur
  - keine Beschreibung der vorgeschriebenen Typen

6

Mediatorsysteme

- Ein Mediator vermittelt zwischen Benutzern und Datenquellen (Middleware)
- Ein Mediator verwendet Wrapper, die ihm homogenen Zugriff auf heterogene Quellen bieten



Mediatoren können hierarchisch angeordnet sein. Mediator enthält Wissen über die zur Verfügung stehenden Datenquellen.

Jede verwendete Datenquelle ist einem Wrapper zugeordnet.

Quellen können DB, Suchmasken, Suchmaschinen oder einfach Webseiten sein.

- Daher: Schnittstelle Wrapper/Mediator muß definiert sein. Das führt zu Notwendigkeit eines Datenaustauschformates für Metadaten, Inhalte, bzw. am besten beidem zusammen. Notwendigkeit von selbstbeschreibenden Daten. Ein frühes System: TSIMMIS

7

Austauschformate

- Datenaustausch im B2B Bereich
  - zB Automobilzulieferer, Austausch von Pressenachrichten
  - Serialisierung von strukturierten Daten
  - Konvertierung in einfach verschickbaren und leicht vom Empfänger rekonstruierbaren Bytestream
- Frühere Formate
  - OEM: Object Exchange Model
  - NetCDF: für mehrdimensionale Array-Daten, aber auch für relationale Daten; generelles Format
  - ASN.1 wird hps. für bibliographische und genetische Daten verwendet
  - ACeDB für genetische Daten, hat starke Ähnlichkeiten mit OEM
  - properitäre Formate wie plain text mit Beistrichen etc.

8

SSD Beispiel

```
{name: "Gottlob", tel: 18420, email: "gottlob@dbai.tuwien.ac.at"}
```

Menge von Paaren mit Bezeichnungen (Label) und Werten.

```
{name: {first: "Georg", last: "Gottlob"}, tel: 18420, email:
   "gottlob@dbai.tuwien.ac.at"}
```

Werte selbst können weitere Struktur beherbergen.

```
{name: {first: "Georg", last: "Gottlob"}, tel: 18420, tel: 18403, email:
   gottlob@dbai.tuwien.ac.at}
```

Wir können auch nicht-eindeutige Bezeichnungen erlauben.

Extensible Markup Language

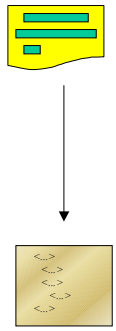
Anforderungen:

- genormte erweiterbare Auszeichnungssprache (W3C)
- Syntax zur Beschreibung (semi)strukturierter Information
- bereichsspezifische Dokumenttypen, Austauschformat
- aber auch Anwendungen und Datenmodellierung
- Trennung von Struktur und Präsentation
- Skalierbare Informationsrepräsentation

"XML will be the ASCII of the Web – basic, essential, unexciting" (Tim Bray)

Anwendung von XML

- Dokument-Anwendungen
  - "menschlicher" Informationsaustausch, B2B, B2C
  - reine Strukturbeschreibung generiert transportables und leicht wiederverwendbares Dokument; einfacherer Informationsaustausch
  - verschiedenste Konvertierungen durch Stylesheets; z.B. einfachere Aufrechterhaltung großer Websites die für verschiedene Browser optimiert sind



- Daten-Anwendungen
  - einfacher "maschineller" Informationsaustausch mit der selben Technologie
  - *"application as document"*
  - automatisierter Datenaustausch mit Datenbanken und Clients
  - XML Datenbanken

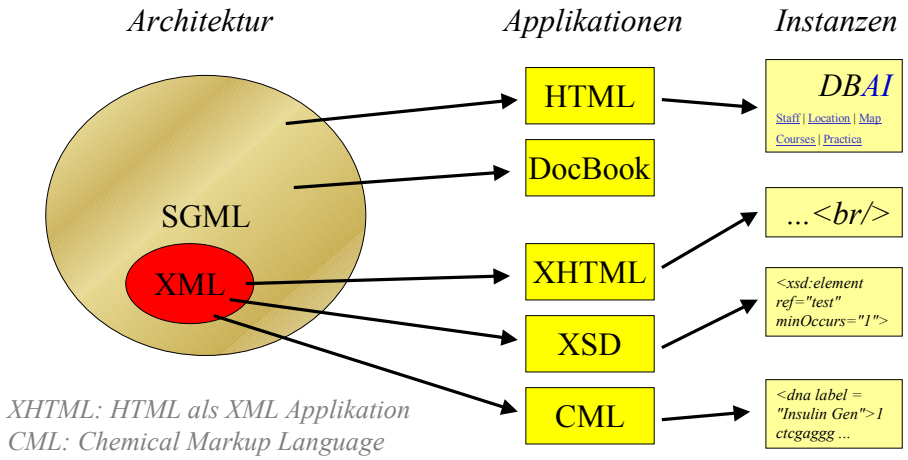
11

HTML vs XML

- HTML
  - Hypertext Markup Language
  - eine Applikation von SGML (eine fixe DTD)
  - HTML 4.0, CGI, Javascript, Flash,....
  - über 100 fixe tags
  - Browser sehr fehlertolerant (ignoriert DTD....)
  - Präsentation (z.B. boldfaced, rot) und Struktur (z.B. Tabellen, Listen)
  - Chaos: verschiedenste properitäre Erweiterungen
  - Semantische Information nur in Metatags
- XML
  - eXtensible Markup Language
  - eine Teilmenge von SGML
  - Metasprache für Markup Sprachen
  - keine prädefinierten Tags
  - strikte Syntax muß eingehalten werden
  - Möglichkeiten Elemente zu vergleichen
  - Abfragesprachen
  - viele ergänzende Standards die in Folge vorgestellt werden
  - leicht zu lesen und zu verarbeiten

12

Applikationen und Instanzen



XHTML: HTML als XML Applikation
CML: Chemical Markup Language
XSD: XML Schema Description (siehe später)

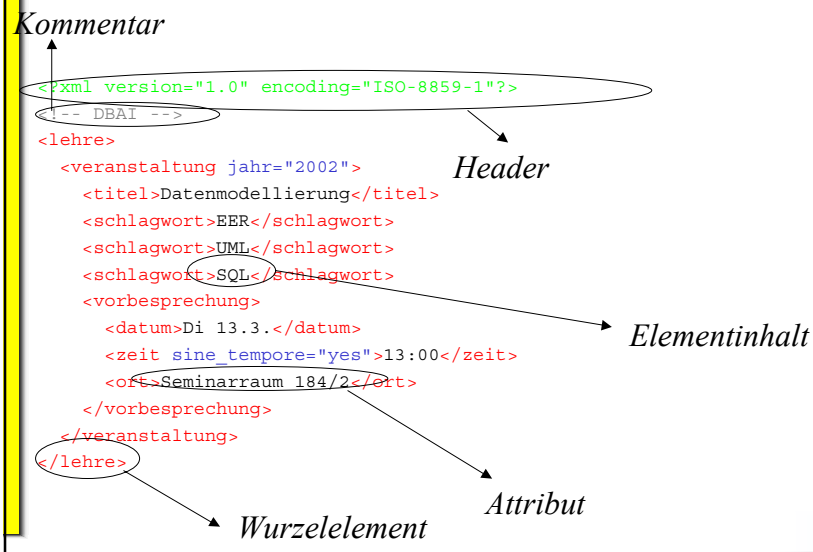
13

Beispiel: XML Dokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DBAI -->
<lehre>
  <veranstaltung jahr="2002">
    <titel>Datenmodellierung</titel>
    <schlagwort>EER</schlagwort>
    <schlagwort>UML</schlagwort>
    <schlagwort>SQL</schlagwort>
    <vorbesprechung>
      <datum>Di 13.3.</datum>
      <zeit sine_tempore="yes">14:00</zeit>
      <ort>Seminarraum 184/2</ort>
    </vorbesprechung>
  </veranstaltung>
</lehre>
```

14

Beispiel: Dokumentaufbau



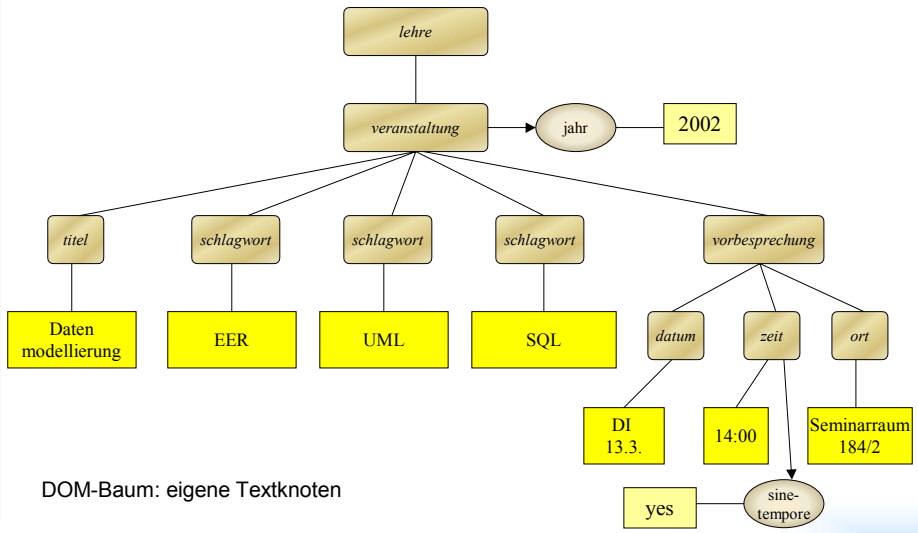
15

Beschränkungen

- Genau ein Wurzelelement pro XML Dokument
  - Name nicht vorgegeben
  - Bei Vereinigungen von Dokumenten: neuen Wurzelknoten hinzufügen
  - auf Ebene von Wurzelelement sonst nur Kommentare/PIs erlaubt
- Endtags
  - jedes Starttag muß geschlossen werden
    Beispiel: `<br>` alleine nicht erlaubt (muß in XHTML `<br/>` lauten)
  - keine verschränkten Tags, eindeutige Kinder/Geschwisteridentifikation
    Beispiel: `<b>bold<i>bold-italic</b>italic</i>` ist nicht erlaubt!
    Aber erlaubt: `<tr>AK der <i>IK</i> 3</tr>`
- Attributeinschränkungen
  - ein Name nur einmal pro Element
    Beispiel: `<preis währung="$" währung="€">` nicht erlaubt

16

Beispiel-Elementbaum



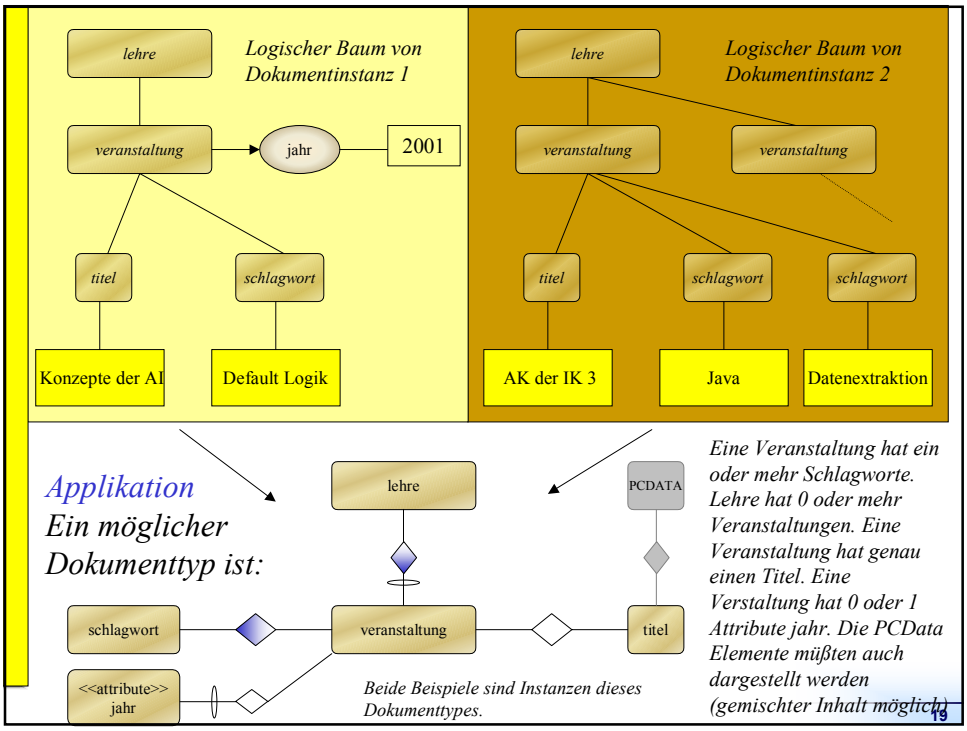
DOM-Baum: eigene Textknoten

17

Kurzbeschreibung: DOM vs. SAX

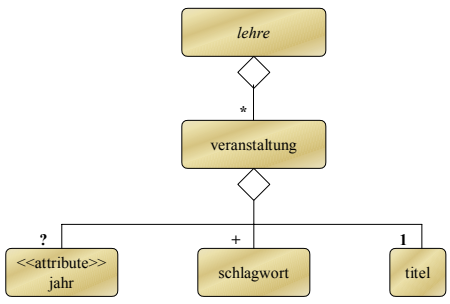
- Document Object Model
  - DOM ist ein programmiersprachenneutrales Objektmodell und Anwendungsprogrammierschnittstelle
  - beschreibt die in einem Dokument einer bestimmten XML-Anwendung enthaltenen Elemente als Objekte für die Verarbeitung mit einer objekt-orientierten Programmiersprache wie z.B. Java.
  - DOM liefert eine komplette Baumstruktur aller Objekte eines XML-Dokuments
  - eignet sich nicht für extrem große XML-Files. Gut für Forms/Editors.
- Simple API for XML
  - Programm-Schnittstelle für die Verarbeitung einer XML Applikation (Klasse von XML-Dokumenten) mit Hilfe einer objekt-orientierten Programmiersprache wie z.B. Java.
  - SAX liefert ein XML-Element nach dem anderen in einem Eingabestrom und eignet sich daher auch für sehr große XML-Files. Gut für App2App Austausch.
- Ein XML Parser liest ein XML Dokument ein und gibt es in Form von DOM oder einer eigenen Struktur mit SAX-Ereignissen wieder. Ein validierender Parser prüft gegenüber einer DTD/XSD.

18



Instanz und Applikation: DTDs

- DTD beschreibt den akzeptierten Dokumenttyp
- DTD z.B. konstruieren aus UML-Modellierung oder darstellen als kontextfreie Grammatik
- einfachere Darstellung des Bsp. von letzter Seite:



20

Beispiel-DTD

```
<!ELEMENT lehre (veranstaltung+)>
<!ATTLIST veranstaltung jahr NMTOKEN #REQUIRED>
<!ELEMENT veranstaltung (titel | schlagwort | vorbesprechung)*>
<!ELEMENT titel (#PCDATA)>
<!ATTLIST titel sprache CDATA "deutsch">
<!ELEMENT schlagwort (#PCDATA)>
<!ELEMENT vorbesprechung (datum, zeit, ort)>
<!ELEMENT zeit (#PCDATA)>
<!ENTITY % boolean "(yes|no) 'no'">
<!ATTLIST zeit sine_tempore %boolean;>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT datum (#PCDATA)>
```

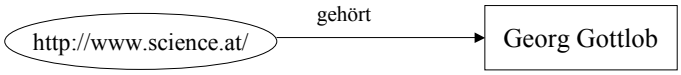
21

XML Schema

- XML Schema Description: ist als XML Dokument dargestellt
  - löst DTDs ab
  - es gibt auch DTD für XSD
- XSD bildet eine vollständig in XML-Syntax formulierte kontextfreie Grammatik zur Formulierung beliebiger XML Strukturen ab.
- Entitäten und Notationen werden nichtmehr auf DTD-Art ausgedrückt
- Strukturen
  - zur Definition von Inhaltsmodellen für Elemente, Attribute und wiederverwendbare Strukturen
  - XML Namensräume werden explizit berücksichtigt
  - spiegelt "die bekannte Mächtigkeit" von DTDs wieder
- Datentypen
  - eigenständiges Typsystem
  - Aufbauend auf diversen Typsystemen

22

Semantic Web: RDF



www.science.at (Subjekt) gehört (Prädikat)
Georg Gottlob (Objekt).

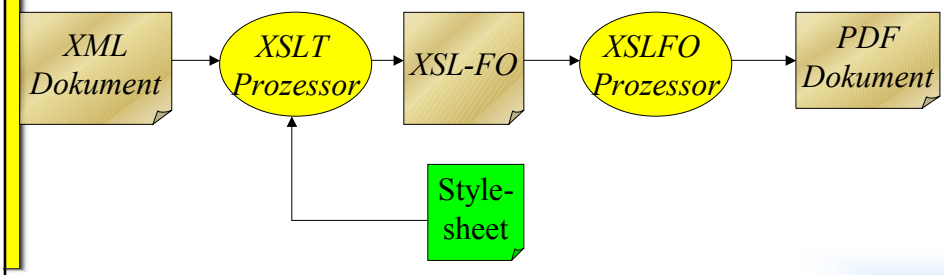
```
<rdf:RDF>
        <rdf:Description about="http://www.science.at">
                <dbai:gehört>
                        Georg Gottlob
                </dbai:gehört>
        </rdf:Description>
</rdf:RDF>
```

"gehört" könnte auch eine Resource sein mit weiteren Eigenschaften.

23

XML Dokument Transformation mit XSL

- Transformation: XSLT benötigt XSL Prozessor
  - z.B. IE5, Xalan, MS-XSL, XT
  - Transformation des Input Dokumentes
  - In Praxis: meist ohne XSL:FO zur Erstellung von HTML, WML, SVG etc.
- Formatierung: XSL:FO benötigt FO Prozessor
  - Apache FOP für PDF Generierung
  - Formatieren des transformierten Dokumentes



24

XSL Beispiel

Input:

```
<lehre>
   <veranstaltung>Datenmodellierung</veranstaltung>
</lehre>
```

Stylesheet:

```
<xsl:template match="/lehre/veranstaltung">
   <i><xsl:value-of select="."/></i>
</xsl:template>
```

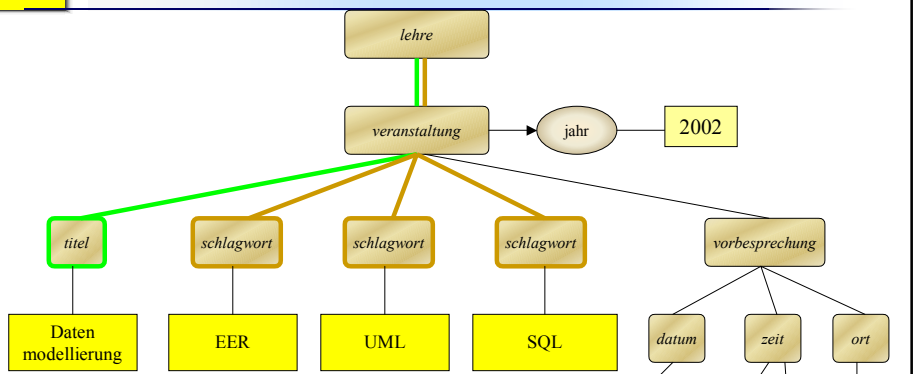
Resultat:

```
<i>Datenmodellierung</i>
```

→ Browser: *Datenmodellierung*

25

XML Dokumentnavigation: XPath



gesamtes Dokument:

/ geht zu Wurzel vor Dokument-Element

`/lehre` *selektiert Dokument-Element*

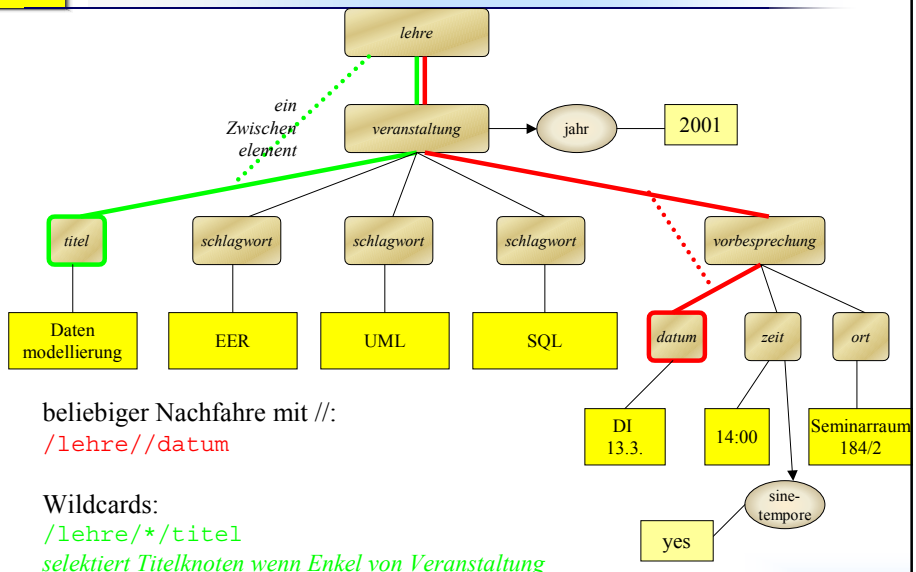
absoluter Pfad:

```
/lehre/veranstaltung/titel
/lehre/veranstaltung/schlagwort
```

gibt Menge von Titelknoten (hier: einen) bzw. Menge von Schlagwortknoten zurück

26

XPath: unvollständige Pfade



beliebiger Nachfahre mit //:

```
/lehre//datum
```

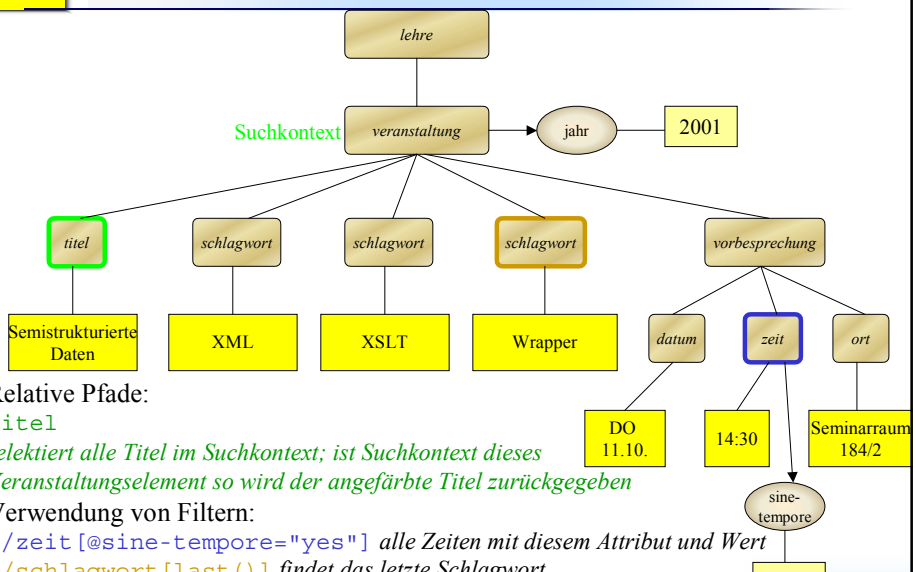
Wildcards:

```
/lehre/*/titel
```

selektiert Titelknoten wenn Enkel von Veranstaltung

27

XPath: Filterdefinitionen



Relative Pfade:

```
titel
```

selektiert alle Titel im Suchkontext; ist Suchkontext dieses Veranstaltungselement so wird der angefärbte Titel zurückgegeben

Verwendung von Filtern:

```
//zeit[@sine-tempore="yes"]
```
alle Zeiten mit diesem Attribut und Wert

```
//schlagwort[last()]
```
findet das letzte Schlagwort

"//" gibt an, daß Element in beliebiger Suchtiefe von Wurzel aus gefunden wird

28

Abfragesprachen

Anforderungen:

- objekterhaltende Queries
- stark getypt im Vergleich zu XSLT
- Flexiblerer Umgang mit well-formed Dokumenten, Unterstützung valider Dokumente
- Flexible Queries auf mehrere Dokumente
- Aufgaben: extract (from), match(where), clip (select), construct (create; Restrukturierung, View)
- Deklarativität
- Standard XQuery
  - http://www.w3.org/TR/query-datamodel/

```
FOR $m IN document("manufacturer.xml")//manufacturer,
    $r IN $m/model/rank
WHERE $r LEQ 10
RETURN $m
```

29

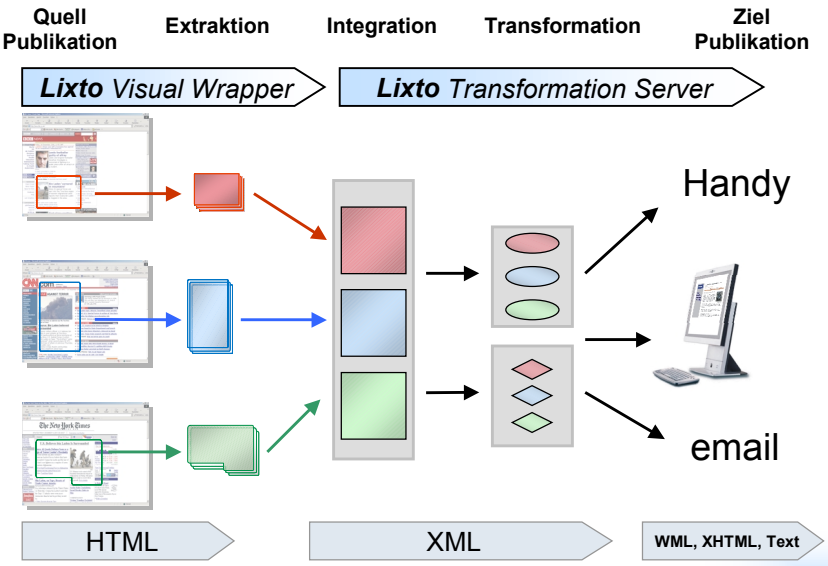
Teil 2

Projekt Lixto

Web Datenextraktion und -integration

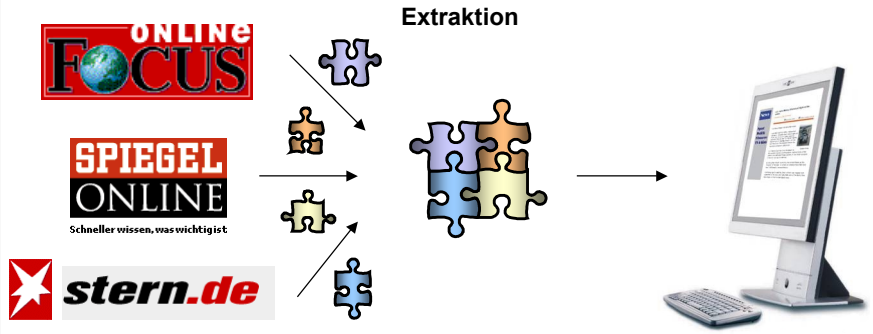
30

Lixto Prozeß Modell



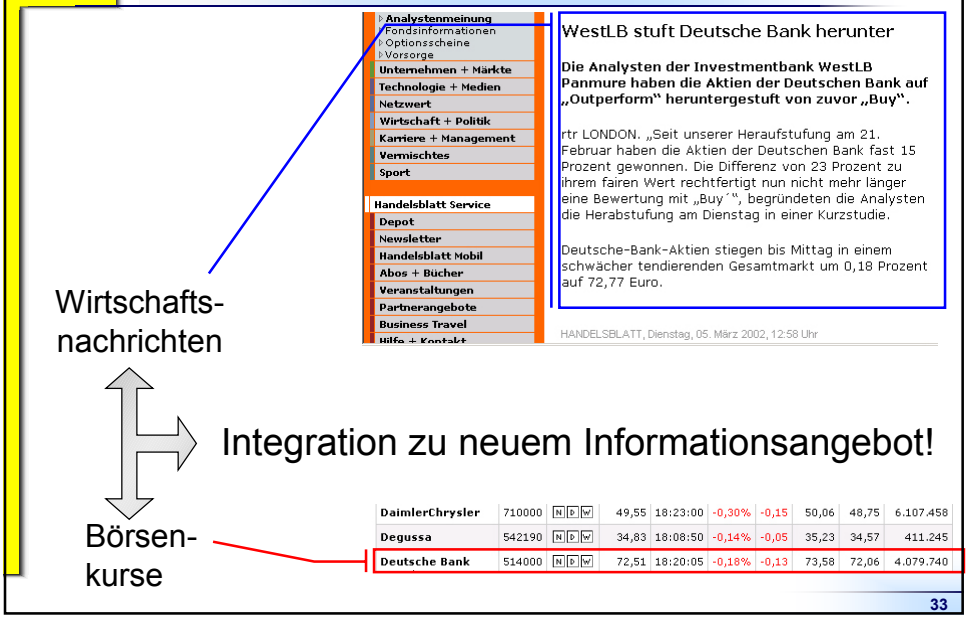
31

Informationsextraktion und -aufbereitung

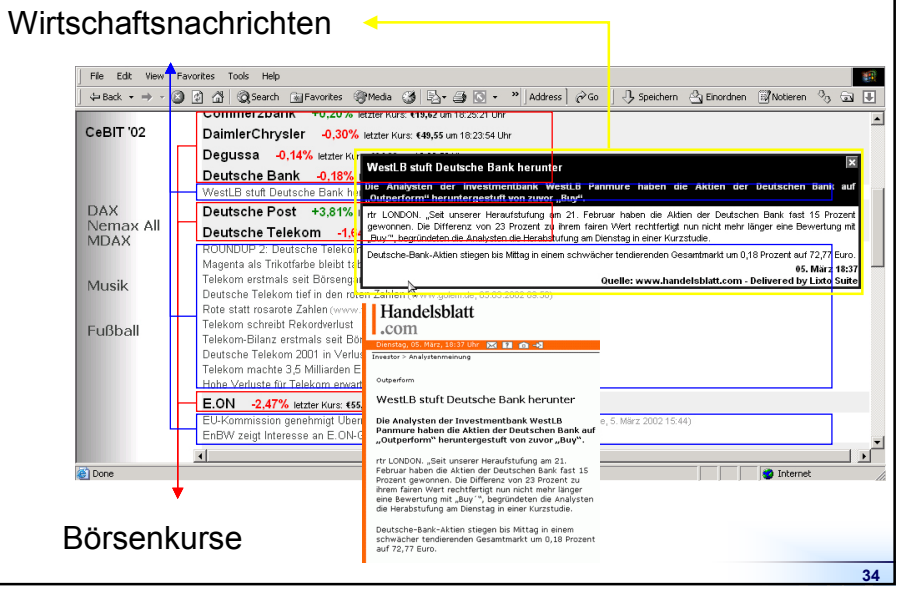


32

Extraktion und Integration



Automatische Publikation



Alternative Endgeräte

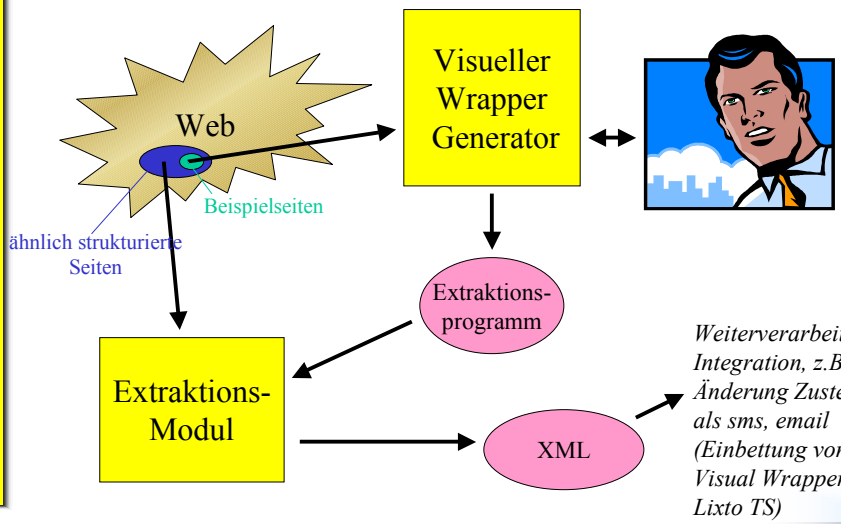
35

Tool-Unterstützung: lixto Visual Wrapper

- Visuelle Definition der Auswahl durch Maus-Clicks
- Präzise Selektion von Teilinhalten (z.B. Überschriften oder Datum) anhand von Strukturmerkmalen
- Ausgabe der Inhalte im XML-Format durch generiertes Wrapper-Programm

36

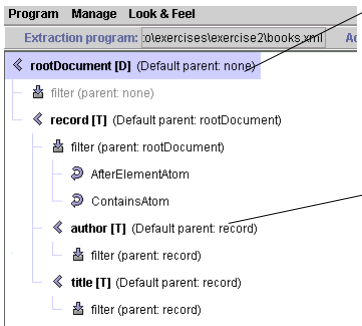
Visual Wrapper Architektur



37

Patternerstellung

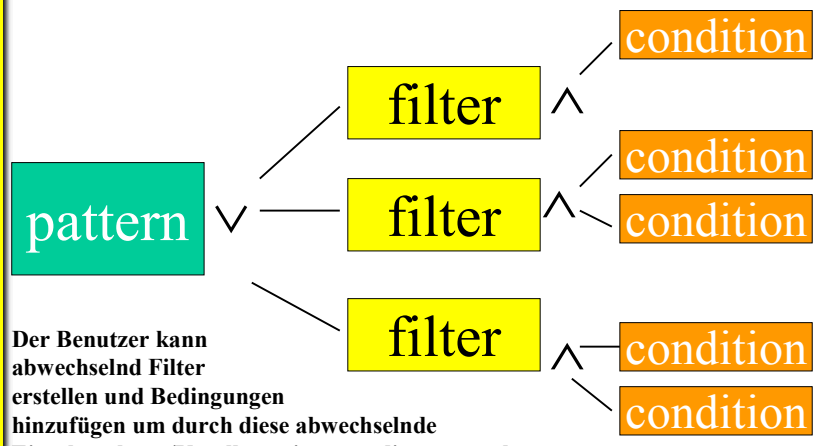
- Neues Programm erstellen
  - Programm ist "hierarchische Ansammlung von "**Patterns**".
- Öffnen einer Beispielseite im Lixto Browser



- Erstellen eines ersten Patterns
  - Ein Pattern charakterisiert eine Art von Information
  - **Wurzelpattern** "rootDocument" hat anfänglich das Beispieldokument als einzige Instanz
  - Die **Instanzen** eines Patterns sind innerhalb der Beispielseiten (z.B. alle Autoren).
  - Der **Name** eines Patterns kann frei gewählt werden. Wird als Defaultname der XML Übersetzung gewählt

38

Patternstruktur



Der Benutzer kann abwechselnd Filter erstellen und Bedingungen hinzufügen um durch diese abwechselnde Einschränkung/Verallgemeinerung die gewünschte Information zu charakterisieren

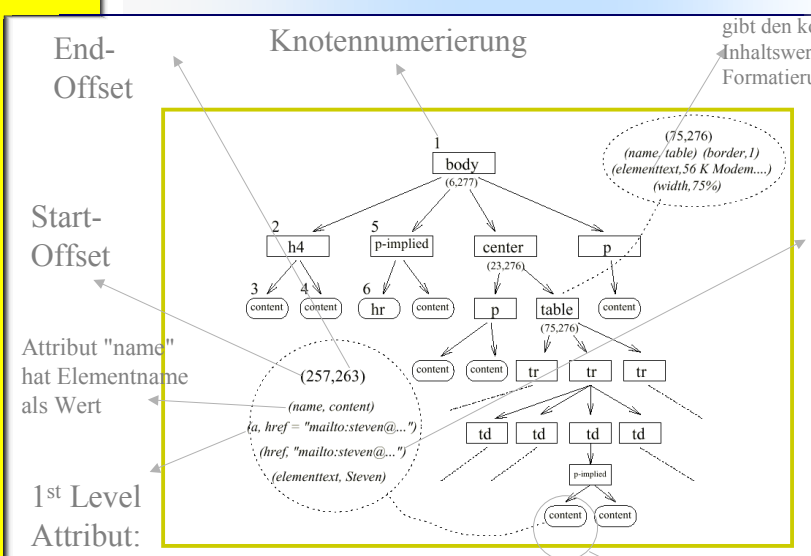
39

Bestandteile eines Filters

- **Verallgemeinerter Pfad**
  - Benutzer markiert ein postives Beispiel auf der Beispielseite
  - System identifiziert den Knoten des HTML Baumes
  - System identifiziert Baumpfad, z.B. *body(1).center(1).table(3)*
  - System verallgemeinert Baumpfad, z.B. **.body.*.center.*.table*
- **Einschränkende Bedingungen**
  - **Attribute einschränken** (enthält, erfüllt ein Konzept, regulären Ausdruck)
  - **Kontextuelle** Bedingungen (before, after, notbefore, etc.)
  - **internal** Bedingungen (contains, firstchild, lastchild)
  - **Range** Bedingungen (z.B. das vierte bis zum vorletzten)
  - **Pattern Referenzen** (Referenz auf zuvor erstellte Patterns)

40

Dokumentmodell



41

Lixto Interne Wissensdarstellung

ELOG: ein Datalog-Dialekt

Wrapper : Elog Programm
Pattern : IDB-Predicate
Filter : Regel
Condition : Atom des Regelkörpers
Parent Pattern : Spezielles Körperatom
Element Pfad : Konstante

Die deklarative und/oder Semantik von Datalog paßt gut zum visuellen Einschränkungs- und Erweiterungsprozeß des Patterngenerierungsprozesses

42

Syntax Elog Regeln

Prädikat das auf ein parent pattern verweist

In Elog werden für Regeln bestimmte Gestalten gefordert.
Die Gestalt der Standardregel ist:

```
NewPattern(S,X) <-  ParentPattern(_,S),
                    ExtrAtom(S,X),
                    Condit(S,X,...)[a,b]
```

Parent Pattern Instanz Variable

Target Pattern Instanz Variable

Je nachdem, ob eine String- oder Baumregel vorliegt, muß X eine Variable sein, die über Baumregionen bzw String Sourcen läuft.

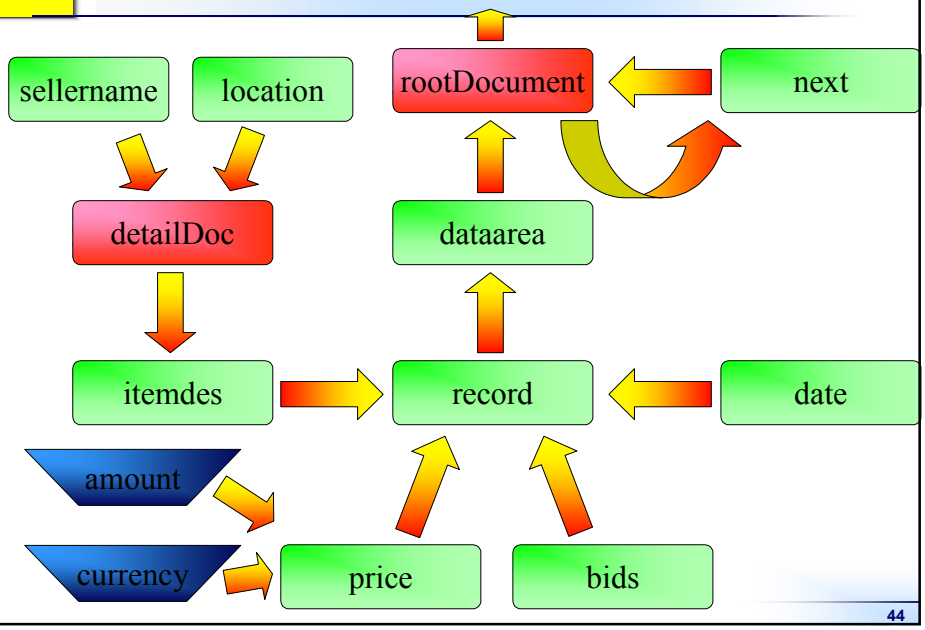
Dokumentregeln haben keine Range-Bedingungen.

Diverse einschränkende Bedingungen je nach Regeltyp, die Instanzen für X aussortieren.

Ein Atom, das festlegt welche Instanzen extrahiert werden. Dieses Atom hat als Inputvariable den Kontext S und gibt Werte für X zurück

43

Patternstruktur eBay



44

Beispiel: eBay XML Output

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
     <record>
          <item>98 Degrees - Notebook - New</item>
          <price>2.99</price>
          <currency>$</currency>
          <bids>-</bids>
          <date>-</date>
    </record>
    <record>
          <item>Notebook - Compaq Presario 1207</item>
          <price>730.00</price>
          <currency>AU $</currency>
```

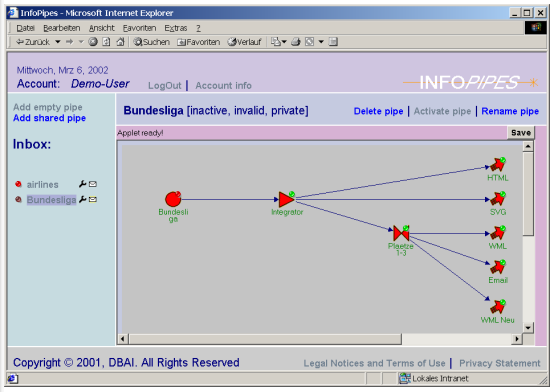
45

Lixto Demo

Lixto Visual Wrapper Demonstration

46

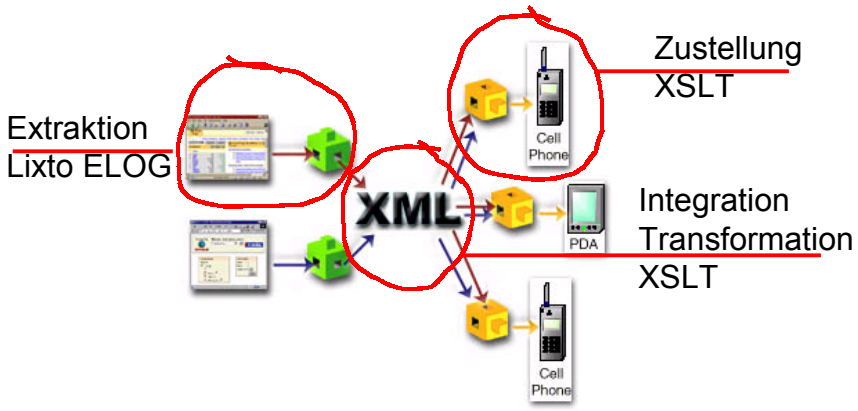
Tool-Unterstützung: lixto Transformation Server



- Definition von Nachrichtenkanälen
- Automatisierte Ausführung der Wrapper-Programme
- Transformation der XML-Daten für die einzelnen Endgeräte

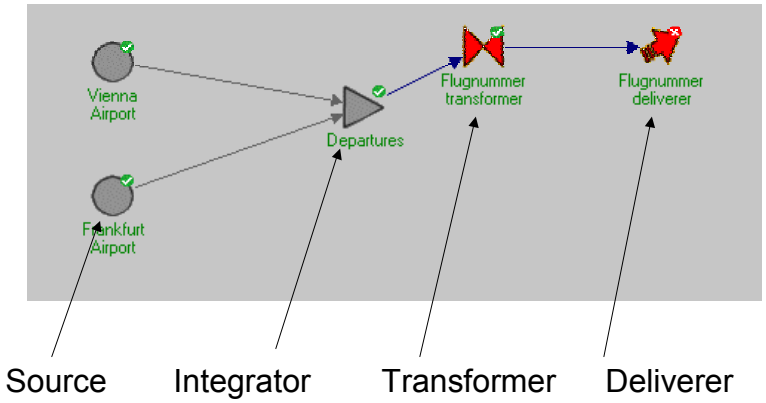
47

Lixto Transformation Server



48

Eine Beispiel Pipe



49

Source Komponente

- Navigation zur gewünschten Webseite
  - Konfiguration durch Beobachtung
  - „deep web“ Extraktion
- Extraktion
  - Inhaltsextraktion -> Inhalt
  - Next button navigieren
  - Auf anders strukturierte Webseiten navigieren

50

Integrator und Transformer

Transformation Server

Integrator

- Einheitlicher View auf eine Anzahl von XML Input Fragmenten
- Input Schema auf Output Schema abbilden
- Syntaktische Abbildung
  - XML elemente wie <e1>,<e2> → <e>
- Semantische Abbildung
  - <e>delay</e>,<e>verspaetung</e> → <e>delay</e>
- Grafisches Interface für XSLT Generierung

Transformer

- Transformiert XML Fragmente
- Joins und Auswahlen
- Grafisches Interface für XSLT Generierung

51

Transformer und Deliverer

Transformation Server

Deliverer

- Information auf multiple Platformen (web, mobil, PDA)
- Push und pull
- Definiert delivery schedule
- XSLT-basiert

Personalisierung

- Publish/Subscribe Mechanismus
- Administrator konfiguriert pipe, definiert Variablen die vom Benutzer gesetzt werden
- System genieriert device-abhängige Userviews
- Benutzer kann pipes subscriben und personalisieren

52

Beispielapplikationen

- Airport Notification [Web, SMS]
  - Monitorieren von Abflugzeiten
  - Informiere User über Veränderungen
- Live Radio Program *NowPlaying* [PDA]
  - Monitorieren von Radio Stationen
  - Anzeigen der momentanen Playlist
  - Benutzer kann Audiostream anhören

53

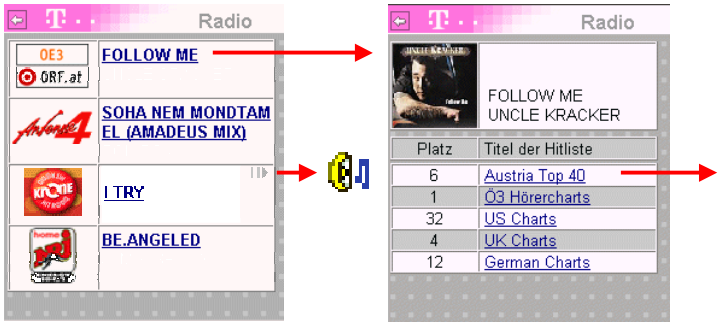
Airport Notifikation

Business Trip Scenario:

- Air traveller hat die folgende Schedule: OS121 12:30 VIE-FRA
  OS128 20:10 FRA-VIE
- Konfiguriert von Administrator
- Personalisation via WEB oder SMS

54

"Now Playing"



55

Lixto Demo

lixto Transformation Server

Lixto Transformation Server Demonstration

56

Verwendete Technologien

- Objektorientierte Programmierung
  - Java: Swing, Servlets, Xerces, ...
- Semistrukturierte Daten
  - HTML, XML, ...
- wissensbasierte Techniken
  - logische Wissensrepräsentation
  - Inferenz
  - Datalog
- Ontologien
- Advanced Usability Model
  - User Centric GUI
  - Java Style Guide

57

Vorteile von Lixto

- Datenextraktion
  - Intelligente Umwandlung von HTML nach XML
  - Interaktive und visuelle Entwicklung von Wrapper-Programmen
  - Robuste und flexible Extraktionsverfahren
  - Ausdrucksfähige interne Sprache
- Integration von Web-Inhalten verschiedener Anbieter
  - Durchgehende Verwendung etablierter XML-Standards
  - Grafische Definition des Transformationsprozesses
- Publikation auf unterschiedlichsten Medien
  - Unterstützung vielfältiger Endgeräte: Handy, (Voice-)Browser, Email
  - Automatisierte Publikationsmechanismen

58

Weiterführende Lehrveranstaltungen

- VU **Semistrukturierte Daten I** (SS)
  - OEM, XML, DTD, XML Schema, XSLT, ...
- VU **Semistrukturierte Daten II** (WS)
  - XSLT, XML APIs, Abfragesprachen (Lorel, XQL, XQuery, ...), Semantic Web, ...
- VU **Datenextraktion und –integration** (SS)
  - XML, Web-Extraktion, Wrappergenerierung, Mediatorsysteme, Aggregation und Syndikation von Daten, Informationskanäle, Portalintegration, ...
- VU **Knowledge Engineering 2** (WS)
  - XML Familie , Java und HTML, Java und XML, Sprachen und Tools zur Datenextraktion, Lixto Projekt, Elog

59

Praktika, Diplomarbeiten

Offene Themen für Praktika und Diplomarbeiten
http://www.dbai.tuwien.ac.at/proj/lixto
Visual Wrapper: baumgart@dbai.tuwien.ac.at
Transformation Server: herzog@dbai.tuwien.ac.at

- Applikationsdesign mit Lixto
- Programmierung am Lixto Core
  - Testsuite
  - Weitere Delivery-Kanäle
  - Adaptive Wrapper
  - Usability Aspekte
  - Mozilla Parser
  - Reusable Rule Library
  - XSD Generierung
  - XQuery Support in Transformation
  - .....
- Technische Benutzerdokumentation

60