Transformation strukturierter Dokumente mit XSLT

Anne Brüggemann-Klein

TU München

Der rote Faden

Überblick und Motivation

Die Sprache XSLT

Pattern von XSLT-Programmen

Ressourcen und Literatur

Wozu dient XSLT ?

XSL Transformations (XSLT) 2.0, W3C Recommendation seit 23. Januar 2007

Programmiersprache zum Transformieren von XML-Dokumenten in andere Formate

- XML → XML (zum Umstrukturieren, als Zwischenschritt)
- XML → (X)HTML (zum formatierten Darstellen im Browser)
- XML → SVG (zur Formatierung einzelner Seiten oder Seitenbereiche)
- XML → XSL-FO → PDF (zur seiten-orientierten Formatierung für Ausdruck)

Beispiel DocBook für technische Dokumentation

- Kodierung in XML
- Anzeige in HTML oder PDF, nach Transformation → CalendarX

Steckbrief XSLT …

XSLT zur Bearbeitung von Dokumentendaten

- Paradigma: Transformation (Querysprache)
  [Pipe-Verarbeitung, angelehnt Publikationskette (Kontext Dokumente)]
  [Abgeschlossenheit, Komponierbarkeit]
- Ursprung: Stylesheets (Teil von XSL = XML Style Language)
- Zielsprachen: HTML, XML (z.B. XSL-FO), Text
- Grundlage: Ausdruckssprache XPath
  - Adressierung von Teilen des XML-Dokuments
  - Konstruktion von XML-Fragmenten durch Berechnung und Transformation
- Grundlage: Datenmodell XDM

Erfahrung: XSLT macht XML operationabel

- ohne SQL sind Tabellen nichts, ohne XSLT (oder XQuery) ist XML nichts

… Steckbrief XSLT

XSLT-Sprache

- deklarativ (regelbasiert), berechenbarkeitsuniversell
- mit XQuery das SQL für XML: XQuery als Abfragesprache, XSLT für Komposition von Ergebnissen und Aufbereitung narrativer Dokumente
  - Drehscheiben-Charakter
  - Cross-Media- und Multi-Channel-Fähigkeit
  - Unterstützung von Single-Source Publishing
- XML-Syntax

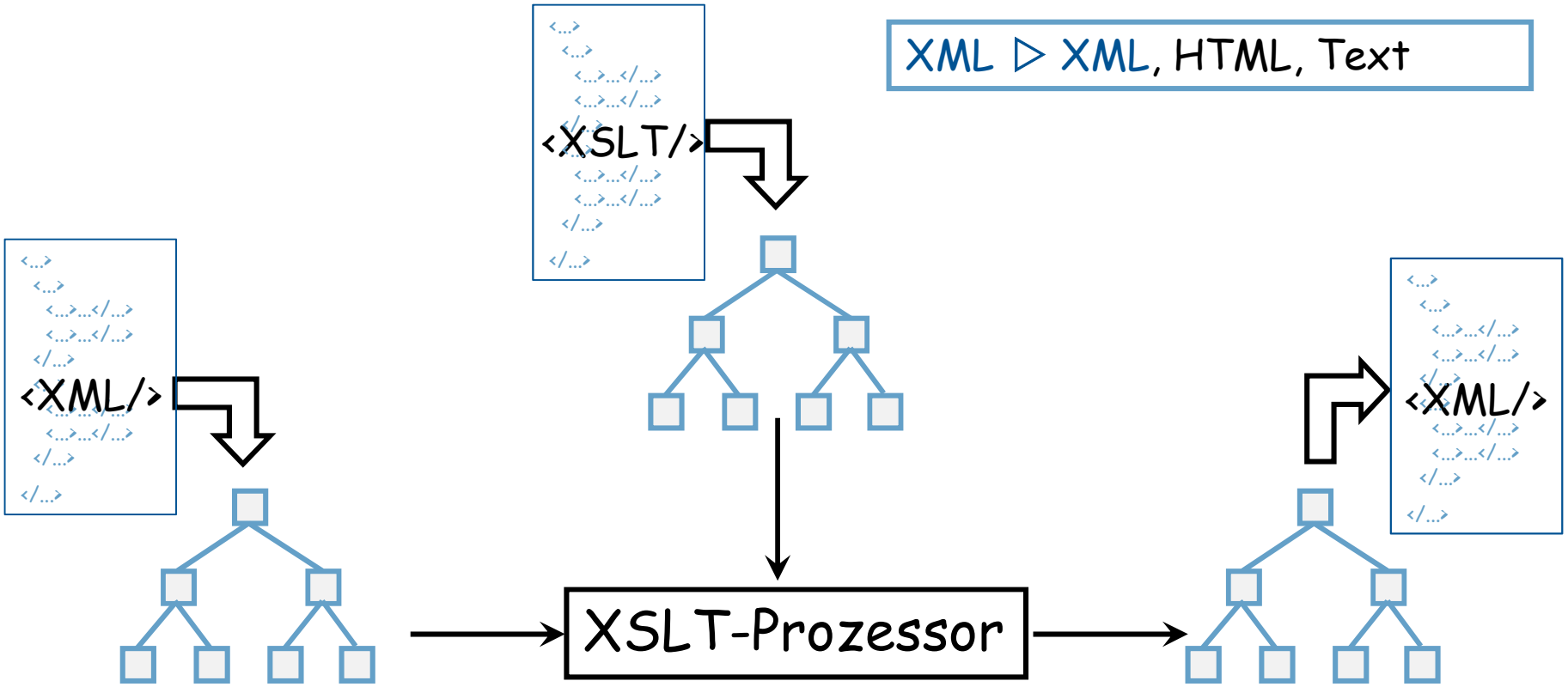
XSLT-Prozessor

- Interpreter
- arbeitet auf Datenmodell

Spezialisierung auf XML-Bearbeitung [gegenüber allgemeiner Programmiersprache / XML-API]

XML ▷XSLT XML

Transformation zwischen Formaten bzw. Baumstrukturen, basierend auf Datenmodell XDM (Information Set)



Die Sprache XSLT

Basiskomponente Template

Basiskomponente eines XSLT-Programms: Template vergleichbar mit Prozedur

Template als Blaupause / Stempel für Ergebnis einer Teil-Transformation

- gegeben als XML-Struktur
- enthält „Freistellen“: Import von Inhalten aus Quelldokument über XPath-Ausdrücke
- bedingter Import möglich
- iterierter Import möglich, definiert durch XPath-Ausdrücke über Eingabedokument
- enthält zusätzlich "Aufhänger", d.h. Angabe wann anwendbar (Attribut match)
- Aufhänger für rekursive Aufrufe weiterer Templates (xsl:apply-templates)
  - parametrisierbar
- Aufhänger für direkten/prozeduralen Aufruf eines Templates (xsl:call-template)
  - parametrisierbar

Einstieg am Beispiel: Pull und Push

xsltBeispiele/Library

Pull: ein einziges Template, das die Ergebnisse zusammensetzt

- zur Umformung von Daten
- Anwendung: Peanuts Pull

Push: spezialisierte Templates for verschiedene Arten von Eingabe-Elementen mit explizit gesteuertem Kontrollfluss (<xsl:apply-templates>)

- Anwendung: Peanuts Push

XSLT-Befehle in Templates …

Wörtliche Daten

- Text (alternativ mit XSLT-Kommando `xsl:text` wegen WS)
- *literal result elements,* auch mit Attributen

XSLT-Kommandos (XSLT-Namespace) für eingabeabhängige und berechnete Daten

- Import von Text aus Eingabedokument:
  `xsl:value-of` mit Attribut `select` (XPath-Ausdruck)
- Import von Teilstrukturen:
  `xsl:copy-of` mit Attribut `select` (XPath-Ausdruck)
- Konstruktion von Elementen und Attributen
  `xsl:element` mit Attribut `name` (XPath-Template)
  `xsl:attribute` mit Attribut `name` (XPath-Template)

… XSLT-Befehle in Templates

XSLT-Kommandos (XSLT-Namespace) zur Ablaufsteuerung

- Bedingte Anweisung:
  `xsl:if` mit Attribut `test` (XPath-Ausdruck)
- Alternative:
  `xsl:choose` mit Kindelementen `xsl:when` und `xsl:otherwise`
- Schleife:
  - `xsl:for-each` mit Attribut `select` (XPath-Ausdruck)
    iteriert über Input-Dokument
  - `xsl:for-each-group` seit XSLT 2.0
    nutzbar für Gruppierung

Verknüpfung von Templates

Transformationsprogramm als Sammlung von Templates
(z.B. pro Element) mit Ergebnis-Skelett

„Freistellen“ im Ergebnis-Skelett füllbar durch rekursiven Aufruf weiterer Templates, gezielt für weitere Elemente relativ zum gerade bearbeiteten "Kontextknoten", eventuell abhängig von Bedingungen oder eingebettet in Schleifen für Wiederholungen

Beim Aufruf eines Templates mit `xsl:apply-templates`

- Angabe welche Elemente bearbeitet werden
- Suche nach passendem Template für jedes Element

Pattern von XSLT-Programmen

Das Push-Prinzip in Reinform

Definition des Push-Prinzips in Reinform

- Ein Template für `/`
- Ein Template pro Elementname
- Jedes Template schafft Ergebnis-Skelett und ruft darin an einer Stelle `<xsl:apply-templates>` für die Kinder auf
- Unmodifizierte Übernahme von Textknoten

Depth-first Navigation durch Eingabedokument

Leichte Erweiterbarkeit

Ähnlichkeit von Eingabe- und Ausgabestruktur

Eignung für narrative, textorientierte Dokumente

Im Wesentlichen angewendet in Peanuts Push (xsltBeispiele/Library)

- Ausnahme: Behandlung des Attributs isbn

Allgemeines Push-Prinzip

Attribut select bei `<xsl:apply-templates select="">` zur gezielten Steuerung der Rekursion, über *Depth-first traversal* hinaus [Genaueres regelt das Prozessmodell]

Mehrere Durchgänge durch ein Dokument möglich durch

- mehrere Aufrufe von `<xsl:apply-templates select="">` in einem Template
- Verwendung von Modi bei `<xsl:apply-templates select="mode">` zum Aufruf von `<xsl:template mode="">`

Drei XSLT-Programmierstile

Pull-Schema (*Fill in the blanks*, navigierend)

- Zielstruktur wird statisch zusammengesetzt, mit Bausteinen aus Quellstruktur
- Mittel: xsl:value-of und prozedurale Anweisungen
- typisch für datenorientierte Anwendungen

Push-Schema (regelbasiert)

- Transformation von Zielstruktur aus Quelle
- Mittel: xsl:apply-templates, Modi
- typisch für textorientierte Anwendungen

Funktionale Programmierung (rekursiv)

- benannte und über Namen aufrufbare Templates mit Parametern
- Turing-Vollständigkeit, Berechenbarkeits-Universalität

Pipes and Filters für XSLT-Programmierung

XML ▷XSLT HTML, XSL-FO / PDF am Beispiel

Komplexere Form der XSLT-Programmierung mit Vielzahl von Regeln, Rekursion, XPath-Ausdrücken, sequentielle Anwendung (xsltBeispieleBook)

Spielzeugbuch über Compilerbau:	compBookNormalized.xml
• Default-Regeln:	bookDefaults.xsl
• Nummerierung:	bookNumbering.xsl
• IDs für Fußnoten:	bookIDFns.xsl
• Verschieben der Fußnoten:	bookMoveFns.xsl
• Auflösen von Referenzen:	bookResolveRefs.xsl
• Identifiziere Kontextabhängiges:	bookMarkContexts.xsl
• Transformiere nach HTML, XSL-FO:	book2htmlEssentials.xsl; book2pdfEssentials.xsl
Ergebnis der Transformation:	compBookFinal.htm, compBookFinal.fo / compBookFinal.pdf

Ressourcen und Literatur

Implementierung

Referenz-Implementierung Saxon (Michael Kay)
für XQuery (damit auch XPath) und XSLT → Bestandteil von oXygen

Download von Saxonia [http://www.saxonica.com/introducing.html]

In den Classpath: saxon.jar

Aufruf für XQuery/XPath

```
java net.sf.saxon.Query <<query>>, z.B.
java net.sf.saxon.Query query.xq
java net.sf.saxon.Query {doc('a.xml')//p[1]}
```

Aufruf für XSLT

```
java net.sf.saxon.Transform
  -o <<output>> <<xmlInput>> <<xsltProgram>>
```

Literatur

Michael Kay: *XSLT 2.0 and XPath 2.0.* 4nd Edition. Wrox 2008.
Code-Beispiele unter http://www.wrox.com/WileyCDA/

http://www.w3.org/Style/XSL/

http://www.xslinfo.com/ (James Tauber)

http://www.oasis-open.org/cover/xsl.html (Robin Cover)

http://www.xml.com/ (O'Reilly)

http://msdn.microsoft.com/xml/

http://www.mulberrytech.com/xsl/xsl-list