XML – basiertes Publizieren und Visualisieren

Seminar der Arbeitsgruppe Datenbanken
und Informationssysteme im WS 02/03

Steffen Apfel

Agenda

1. Einführung in das Themengebiet
2. XSL : XSL/T & XSL/FO
3. SVG
4. DocBook
5. Single-Source-Publishing: Beispiel
6. Frameworks für Web-basierte Systeme
7. Fazit

Einführung in das Themengebiet

Ziel des XML Publishing und Visualisieren:

Daten in **einem** Dokument speichern und je nach Verwendung in **unterschiedliche** Ausgabeformate transformieren

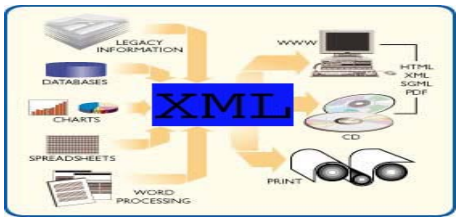
In welchem Format sollen die Daten gespeichert werden ?

SGML: zu komplex und nicht für den Datenaustausch im Web geeignet

HTML: nicht erweiterbar; dient eher der Präsentation, nicht Datenspeicherung

⇨ **XML** bietet einen Kompromiß zwischen Einfachheit und Flexibilität

Einführung in das Themengebiet



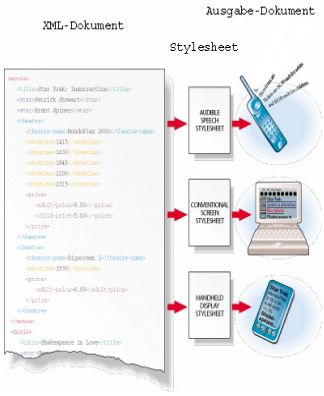
- XML trennt den Inhalt (Daten) von der Präsentation/Layout
- XML hat keinen festen Satz an Markup-Befehlen und kann erweitert werden
- XML-Dokumente kann zum Datenaustausch verschiedener Anwendungen benutzt werden
- XML kann leicht im Netz übertragen werden
- Aus XML kann man alle gängigen Ausgabeformate erstellen
- XML speichert Daten plattformunabhängig und kann sie beliebig kombinieren

Single-Source-Publishing

- **S**ingle **S**ource **P**ublishing (SSP) verfolgt die Idee des „**store once, use everywhere**“
- Man speichert seine Daten *unabhängig* von der *späteren Anwendung* in XML und generiert bei einem konkreten Anliegen erst sein Ausgabeformat
- Modifikationen müssen nur einmal im XML-Dokument vorgenommen werden
- Zeit und Geldersparnis
- Die nun vorgestellten Techniken kommen beim SSP zum Tragen

XSL

- e**X**tensible **S**tylesheet **L**anguage (XSL)
- seit dem 15.Oktober 2001 in der Version 1.0 eine **Recommendation des W3C**
- Mit Stylesheets werden die datenorientierte XML-Dokumente in XML-Dokumente mit **Layoutinformationen** transformiert
- Mit Hilfe der Stylesheets werden die XML-Dokumente in ein **Ausgabeformat** für Browser, Drucker oder PDA´s umgewandelt



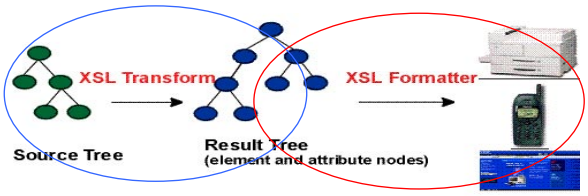
XSL Untersprachen

XSL stehen zur Verarbeitung drei Sprachen zur Verfügung:

- Mit **XPath** (**XML Path Language**) werden bestimmte Teile des XML-Dokuments adressiert und angesprochen. Die „Pfadbeschreibungssprache" wird zum Matching verwendet.
- **XSL/T** (**eXentsible Stylesheet Language Transformation**) beschreibt, wie man die Baumdarstellung eines XML-Dokuments in eine andere XML-Baumdarstellung verwandelt. XSL/T gibt die Regeln zur **Transformation** an.
- Mit **XSL/FO** (**XSL Formatted Objects**) werden die XML-Dokumente **formatiert**. Durch XSL/FO kann man einem XML-Dokument neben dem Inhalt noch Layout-Informationen bei fügen.

XSL Verarbeitung eines XML-Dokumentes

XSL Stylesheet Processor bearbeitet das XML-Dokument in zwei Teilschritten:



PDF, RTF, PS

Datenreduktion bei Anzeige in Handy oder PDA

XHTML für Webdarstellungen

Transformation des Source Tree (Ausgangsdokument in XML) in einen Result Tree (ebenfalls ein XML-Dokument)

XML-Elemente wurden in diesem Schritt in XSL/FO-Elemente transformiert

Einzelne Knoten können dabei umstrukturiert werden

Result Tree wird mit einem **XSL Formatter** in das Ausgangsdokument durch Formationssemantik formatiert

XSL/T

- **XSL/T** (e**X**tensible **S**tylesheet **L**anguage **T**ransformation)
- Version 1.0 seit dem 16.11.1999 eine **Recommendation des W3C**
- XSL/T ist ein wohlgeformtes XML-Dokument mit eigenem Namensraum um XML-Dokumente in andere XML-Dokumente zu transformieren
- **XPath** wird von XSL/T benutzt, um bestimmte Teile eines XML-Dokuments anzusprechen
- Da XSL/T **generisch** wohlgeformte XML-Dokumente in andere XML-Dokumente transformiert, sind die entstehenden Dokumente auch immer wohlgeformt
- Die häufigsten Ausgabeformate sind XML, XHTML und XSL/FO-Dokumente

XSL/T Aufbau eines Stylesheets

```
1: <?xml version="1.0" ?>
2: <xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
3:   <xsl:template match="fussball">
4:         <html>
5:         <xsl:apply-templates />
6:         </html>
7:   </xsl:template>
8: </xsl:stylesheet>
```

XSLT-Stylesheets bestehen aus einer Serie von **Templates**. Jedes Template beinhaltet:

Muster (Pattern), das XML-Elemente im XML-Quelldokument selektiert und auswählt

Körper, der spezifiziert, was für eine Ausgabe generiert werden soll

XSL/T XML & Stylesheet

XML-Dokument:

```
<results group="A">
   ...
<game>
<date>10-Jun-1998</date>
<team score="2">Brazil</team>
<team score="1">Scotland</team>
</ game >
   ...
</results>
```

Stylesheet:

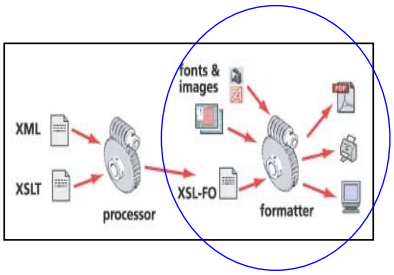
```
<xsl:template match=" game ">
<h2>
      <xsl:value-of select="team[1]" />  versus
      <xsl:value-of select="team[2]" />
      </h2>
      <p>
       Played on  <xsl:value-of select="date" />
      Result:
       <xsl:value-of select="team[1]" />
      <xsl:value-of select="team[1]/@score" /> ,
      <xsl:value-of select="team[2]" />
      <xsl:value-of select="team[2]/@score" />
      </p>
      </xsl:template>
```

XSL-Prozessor parst XML ⇨ Vergleicht Knoten mit Muster ⇨ ersetzt Knoten mit Körperinhalt

XSL/T

XHTML-Dokument nach der Transformation durch ein Stylesheet:

XSL/FO



eXtensible Stylesheet Language Formatted Object (XSL/FO)

Seit dem 15.Oktober 2001 ist XSL/FO eine W3C-Recommendation

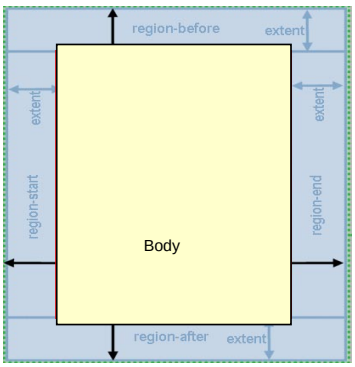
XSL/FO fügt einem inhaltsbasiertem XML-Dokument **Layoutinformationen** hinzu

XSL/FO-Dokumente kann man sich im Browser anzeigen oder automatisch in druckbare Formate umwandeln lassen

XSL/FO kann externe Dateien einbinden

XSL/FO Formatter erzeugt aus den einzelnen Dateien und dem XSL/FO-Dokument **ein** Ausgabedokument

XSL/FO Dokumentaufbau 1.Stufe



Das Tag ***<layout-master-set>*** legt das Layout und den Aufbau *mehrerer* Seiten fest

Eine Masterseite kann in bis zu fünf **Region-Areas** unterteilt werden:

Ein ***Body*** für den eigentlichen Seiteninhalt

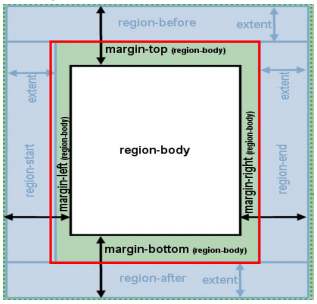
Before, ***After*** für Kopfzeilen und Fußzeilen

Start und ***End*** für Randbemerkungen

→ **Grundgerüst** für den Aufbau aller Seiten (ähnlich zu Masterlayout bei Powerpoint)

XSL/FO Dokumentaufbau 2.Stufe

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="Erste_Seite"
        page-height="29.7cm" page-
        width="21cm"
        margin-top="1cm"
        margin-bottom="2cm"
        margin-left="2.5cm"
        margin-right="2.5cm">
  </fo:simple-page-master>
</fo:layout-master-set>
```



Innerhalb des Master-Tags wird jede einzelne Seite mit dem Tag ***<simple-page-master>*** näher definiert

Die Seite wird in mehrere Block-Areas innerhalb des Region-Bodies unterteilt

In Block-Areas werden Texte, Grafiken oder weitere Formatierungsobjekte eingebunden:

```
<fo:block>
<fo:external-graphic src="bild.jpg"/>
</fo:block>
```

→ Somit kann das Layout einer Seite bis ins kleinste Detail organisiert werden

SVG

- **Scalable Vector Graphics (SVG)**
- Version 1.1 ist seit dem 14.Januar 2003 eine **W3C-Recommendation**
- Auf XML basierende **Vektorgrafiksprache**
- Dank **Xlink** kann jeder Bereich einer SVG-Grafik verlinkt werden
- SVG ermöglicht das Schwenken und **Zoomen** in Grafiken ohne Qualitätsverlust
- SVG-Inhalte können über **XSL** formatiert werden
- Erzeugte Dateien benötigen **weniger Speicherplatz** als entsprechende GIF-, JPG- oder PNG-Dateien
- SVG unterstützt **Animationen** und **Interaktionen** (z. B.: mouseon, onClick)

SVG einzelne Elemente

Rechteck

```
<rect x="5" y="5" rx="5" ry="5" width="40"
height="40" fill="red"/>
```

Optional rx, ry für gerundete Rechtecke.

Text auf einem Pfad


```
<defs><path id="textPath" d="M10 50 C10 0 90 0 90
50"/></defs>
<use xlink:href="#textPath" stroke="blue" fill="none"/>
<text fill="red"><textPath xlink:href="#textPath">Text on a
Path</textPath></text>
```

Filtereffekte:

Durch die Filter wird das Bild gerendert und verändert angezeigt, bleibt aber in seinem Originalzustand

Somit kann das Originalbild bei bleibendem Filter beliebig ausgetauscht werden

SVG Anwendungen

Zoomen in Grafiken:

Zeichenebenen:

DocBook

- **DocBook** Version 5 ist eine Document Type Definition um strukturierte Dokumente in XML zu schreiben
- Seit **1991** wurde DocBook in einem gemeinsamen Projekt von HaL Computer Systems und O´Reilly entwickelt und seit **1998** von OASIS übernommen
- DocBook ermöglicht es dem Autor seine Bücher und Artikel in einer **präsentationsneutralen Form** zu speichern
- mit Hilfe von **XSL** kann man DocBook in andere Formate wie z. B. XHTML, PDF, PS, XSL/FO, man pages (Unix), u.s.w. transformieren

Die Elemente der DocBook DTD werden in zwei Klassen unterteilt:
hierarchischen Elemente und **Info-Pool** Elemente

DocBook hierarchischen Elemente

dienen zur Beschreibung einer umfassenden Struktur bzw. Gliederung des Dokuments in einzelne Teile:

Das **SET** Element enthält mehrere Bücher, welche sich einem Thema widmen

Das **BOOK** Element ist das geläufigste Top-Level Element unter Set für einzelne Bücher.

Das **ARTICLE** Element kann z.B. für Zeitschriftenartikel, White Papers und technische Anmerkungen genutzt werden

REFERENCE PAGES werden für Unix Man-Pages genutzt

Weitere Elemente:
Widmung (**dedication**)
Unterteilungen (**part, reference**)
Komponenten, wie Vorwort (**preface**), Kapitel (**chapter**), Anhang (**appendix**), Glossar (**glossary**)

```
<book>
 <bookinfo>
  <title> Mein Buch </title>
 <author>
  <firstname>Steffen</firstname>
  <surname>Apfel</surname>
  </author>
 <copyright>
  <year>2003</year>
  <holder>AGDBIS</holder>
  </copyright>
  </bookinfo>
 <preface>
  <title>Vorwort</title>
        ...
  </preface>
  <chapter>...</chapter>
  <chapter>...</chapter>
  <appendix>...</appendix>
  <index>...</index>
  </book>
```

DocBook Info-Pool Elemente

Werden nochmals in Block-Elemente und Inline-Elemente unterteilt:

Block-Elemente:

- Listen (itemizedlist, orderedlist, ...)
- Beispiele (example), Abbildungen (figure) und Tabellen (table)
- Absätze (para, ...)

Inline-Elemente:

- traditionelle (footnote, quote, ...)
- Cross Referenz (anchor, link, ...)
- Mathematik (subscript, ...) hierfür besser MathML
- Nutzerinterface (guibutton, shortcut, ...)

SSP Toolvorstellung: AntennaHouse

- **AntennaHouse** bietet eine komfortable grafische Benutzeroberfläche, um XML in XSL/FO umzuwandeln
- Die XSL/FO-Dokumente kann man sich in einem **Browser** anzeigen lassen oder im **XSL Formatter** in PDF umwandeln

XML-Dokument → **XSL/FO-Dokument**

SSP Toolvorstellung: AntennaHouse

XML-Dokument:

```
<?xml version="1.0" encoding="utf-8" ?>
<test>
<section>
<subsection>
<title> XML Publishing </title>
<para>creating a first table</para>
<tbl>
<row>
<col>
<para>Ausgangsdatei: XML</para>
<para>mit Hilfe von XSL</para>
<para>ensteht ein XSL/FO - File</para>
</col>
</row>
</tbl>
<para />
<para>creating a second table</para>
<tbl>
<row>
<col>
<para>Dieses XSL/FO - File kann </para>
</col>
<col>
<para>im Web dargestellt werden</para>
</col>
…
</row>
</tbl>
</subsection>
</section>
</test>
```


SSP Toolvorstellung: AntennaHouse

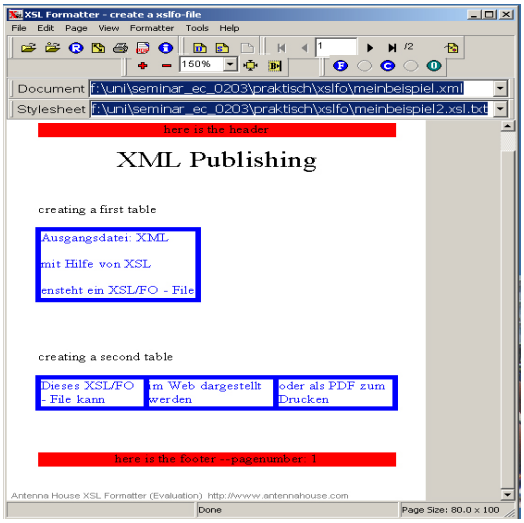
XSL-Dokument:

```
<xsl:template match="/">
  <root font-family="Times" font-size="20pt">
    <layout-master-set>          …          </layout-master-set>
      <page-sequence master-reference="frame" initial-page num="1">
        …
      </page-sequence>
    </root>
  </xsl:template>
```

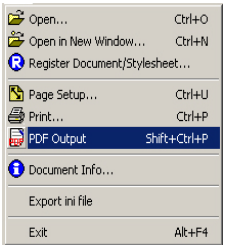
```
<xsl:template match="title">
      <block text-align="center" font-size="2pt" space-after.optimum="40pt">
            <xsl:apply-templates />
      </block>
  </xsl:template>
```

```
<xsl:template match="tbl" name="do-a-table">
    <table color="blue">
      <table-body>
        <xsl:apply-templates />
      </table-body>
    </table>
  </xsl:template>
```

SSP Toolvorstellung: AntennaHouse



XSL/FO kann im **Browser** angezeigt werden oder im **XSL Formatter** leicht in PDF umgewandelt werden



Frameworks für Web-basierte Systeme

Systeme/Frameworks für Web-basierte Systeme stellen eine Plattform für Single-Source-Publishing-Systeme dar

Die Frameworks bieten modulare Schnittstellen, um eigene Erweiterungen einzubringen, so dass einzelne Komponenten ausgetauscht werden können

Stellvertretend für Frameworks für XML-Publishing und Visualisieren wird **Cocoon** von **Apache,** welches seit 1999 entwickelt wurde, nun näher vorgestellt:

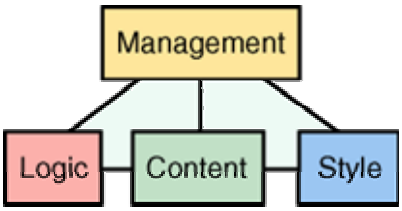
Framework: Cocoon

Aufgabe dieses Publishing Frameworks:

Dynamische Gestaltung von Webseiten und Präsentationen erstellen
Angeforderte Ressource für Nutzung aufbereiten je nach anfragendem Client

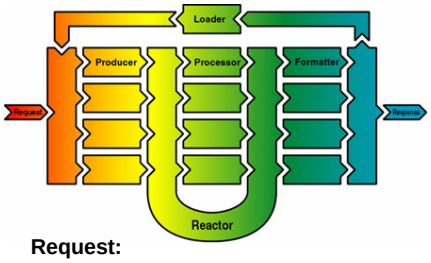
- **Separation of Concerns**, d. h. Trennung von *Inhalt*, *Logik* und *Darstellung* bei der Entwicklung von Webseiten
- **Datenquellen einbinden**: Filesysteme, native XML Datenbanken, und Netzwerkbasierte Quellen
- in **Präsentationsformat überführen** und ausliefern (in XHTML, WML, PDF, SVG, RTF)
- **Dynamischer Webinhalt** mittels XSP

Framework: Separation of Control



- Klare Trennung der verschiedenen Arbeitsbereiche in einem Entwicklungsteam von Webseiten
- **Management**: Abläufe, Aufbau
- **Content**: redaktioneller Teil aller Inhalte
- **Logic**: Integration der dynamischen Inhalte
- **Style**: Präsentation, Grafik und „look & feel“
- Austausch nach bestimmten Regeln möglich, die vom Management gesteuert werden

Framework: Arbeitsweise von Cocoon1

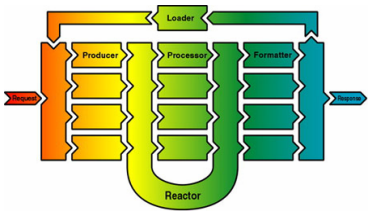


Request:
Der Request enthält alle für die Verarbeitung der Anfrage wichtigen Informationen

Producer:
Ein Producer bearbeitet den eingegangenen Request und liefert als Ergebnis ein XML-Dokument

Reactor :
Der Reactor bestimmt auf Basis von **PI**´s, welcher Prozessor das XML-Dokument weiterverarbeiten soll. Nachdem die Verarbeitung abgelaufen ist, ist es Aufgabe des Reaktors, das überarbeitete Dokument an den vorgegebenen Formatter weiterzuleiten.

Framework: Arbeitsweise von Cocoon1



Processor:
- XSL/T-Prozessor
- XSP-Prozessor
- SQL-Prozessor

Formatter:
Ein Formatter formt das XML-Dokument in einen für Browser/Client verarbeitbaren Datenstrom um (MIME Typ setzen, ausführbarer Code).

Response:
Der Response fasst das neu erzeugte Dokument und schließt den Verarbeitungsprozess ab.

Loader:
Der Loader übergibt Dokumente mit zusätzlichen PI´s wieder an den Producer und gliedert diese wieder in die Verarbeitungskette ein für einen evtl. zweiten Verarbeitungsschritt.

Framework: Neuerungen Cocoon2

Performance-Verbesserungen:

- SAX anstatt DOM → weniger Speicherverbrauch
- Simultane Verarbeitung mehrerer Anfragen
- Skalierbarkeit je nach Lastsituation
- Response wird schon während der Bearbeitung gesendet (schlecht bei inkrementeller Verarb.)
- „Hot Spot“-Erkennung (wieder verwendbarer Kode)
- Carbage Collection wird durch SAX-Verarbeitung geringer
- Kein Reactor-Prinzip mehr (starre Folge von Bearbeitungsschritten) → Pipeline Mapping Technique (flexible Nutzung durch Sitemaps)

Fazit

- **XML liefert flexible Mechanismen für den Datenaustausch und die Aufbereitung von Inhalten für die unterschiedlichsten Zielmedien**
- **XML ist erweiterbar und kann an bestehende Problemlösungen angepasst werden**
- **XML ist außerdem von Mensch und Maschine leicht lesbar**
- **Dieses Seminar sollte dem Leser einen kleinen Einblick in die vielfältigen Verarbeitungs- und Einsatzmöglichkeiten von XML geben**
- **XML verwirklicht das Ziel „Store once, use everywhere“**

Vielen Dank für Ihre Aufmerksamkeit!

Für weitere Fragen stehe ich
Ihnen nun gerne zur Verfügung.