

Verwaltungsaufgaben lösen mit XML und \LaTeX

Uwe Siart
T \E X-Stammtisch München
tutorien@siart.de

Erstellt: 11. März 2003
Zuletzt geändert: 7. November 2008

Listen

- Adressenlisten
- Teilnehmerlisten
- Namensschilder
- Tagungsprogramme
- Ressourcenübersichten

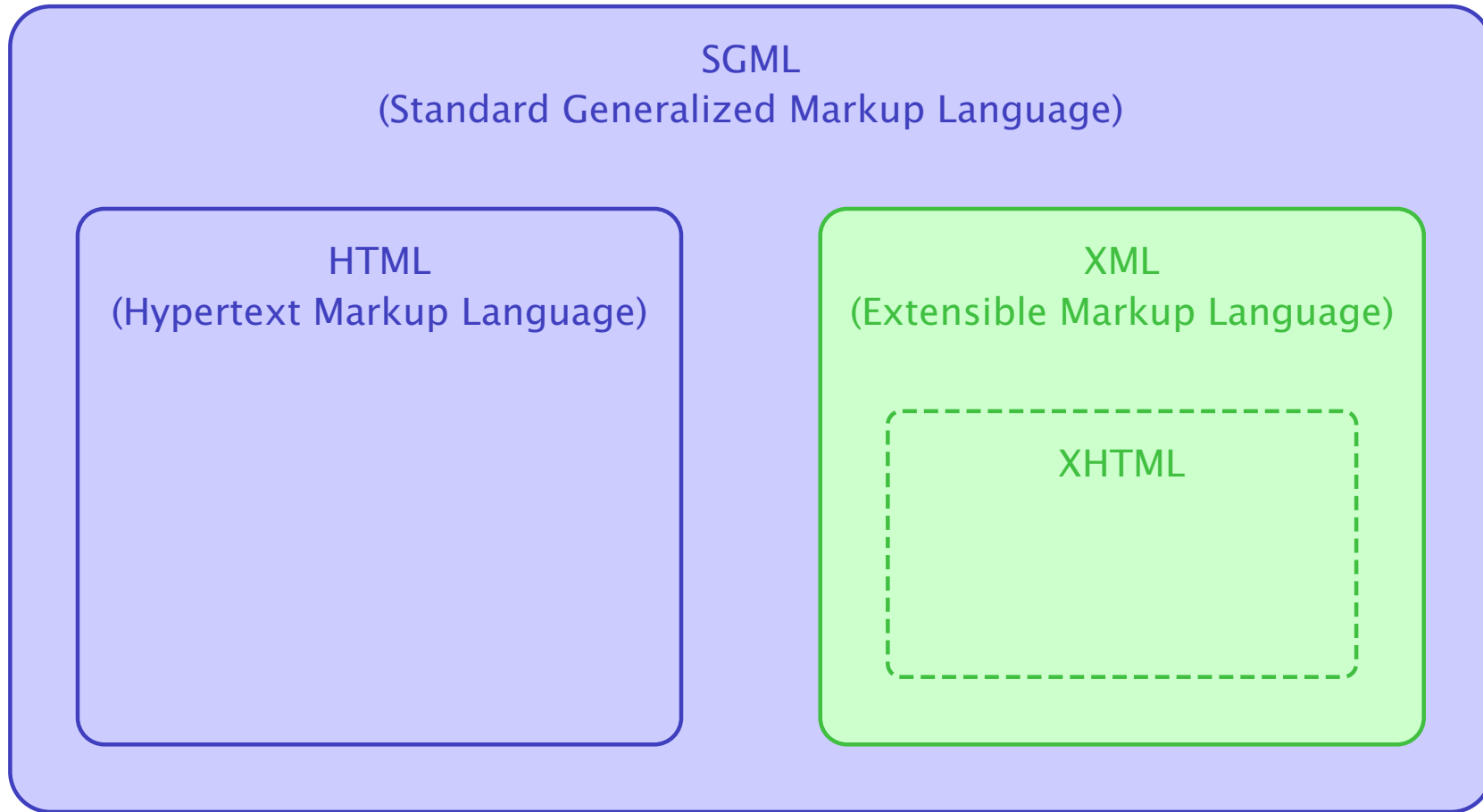
Serienbriefe

- Anmeldebestätigungen
- Rechnungen

Was kann man mit XML und XSLT machen?

- Mit XML können Informationen (Daten) strukturiert und ohne jede Festlegung der Ausgabeform erfasst werden (»Datenbank«). Die Datenstruktur ist hierarchisch (Baumstruktur).
- Mit der Transformationssprache XSLT kann ein Dokumentenrumpf erzeugt werden. Für verschiedene Datenknoten einer XML-Datei geben Vorlagen an, was daraus im Ausgabedokument erzeugt werden soll. Der Zugriff auf die Daten kann dabei auch durch Sortieranweisungen, Bedingungen und Fallunterscheidungen gesteuert werden.

Einordnung der Sprache XML



Eigenschaften der Sprache XML

- Elemente, Attribute und Verschachtelungsregeln frei definierbar
- Ausschließlich *logisches* Markup, die Form der Ausgabe entsteht bei der Verarbeitung
- Verarbeitende Software darf nicht „raten“, was gemeint sein könnte. Bei ungültigen Dokumenten muss die Verarbeitung abgebrochen und ein Fehler ausgegeben werden.
- Ähnlich mächtig, wie SGML, aber weniger häufig benutzte Features fallen weg

XML ist eine Notationsvorschrift zur Beschreibung einer Dokumentenstruktur.

Wohlgeformtheit

- XML-Deklaration am Anfang
- mindestens ein Datenelement
- es gibt ein äußerstes Element (Wurzelelement)
- Baumstruktur (hierarchisches Datenmodell)
- schließende Tags (auch bei leeren Elementen)
- richtige Verschachtelung

Gültigkeit Ein XML-Dokument ist gültig, wenn es wohlgeformt ist und einer DTD (Document Type Definition) gehorcht.

Beispiel für ein XML-Dokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<seminar>

  <teilnehmer bestaetigt="ja" jahr="2005">
    <anrede>Herr</anrede>
    <grad>StR z. A.</grad>
    <vorname>Friedrich</vorname>
    <nachname>Wasweissich</nachname>
    <schule>Hans-Hopp1a-Schule</schule>
    <strasse>Stolperstr. 22</strasse>
    <plz>97777</plz>
    <ort>Fallera</ort>
    <tel>(09999) 3222-0</tel>
    <fax>(09999) 3222-20</fax>
    <email>Friedrich.Wasweissich@account.de</email>
    <notiz></notiz>
  </teilnehmer>

</seminar>
```

Beispiel für ein XML-Dokument

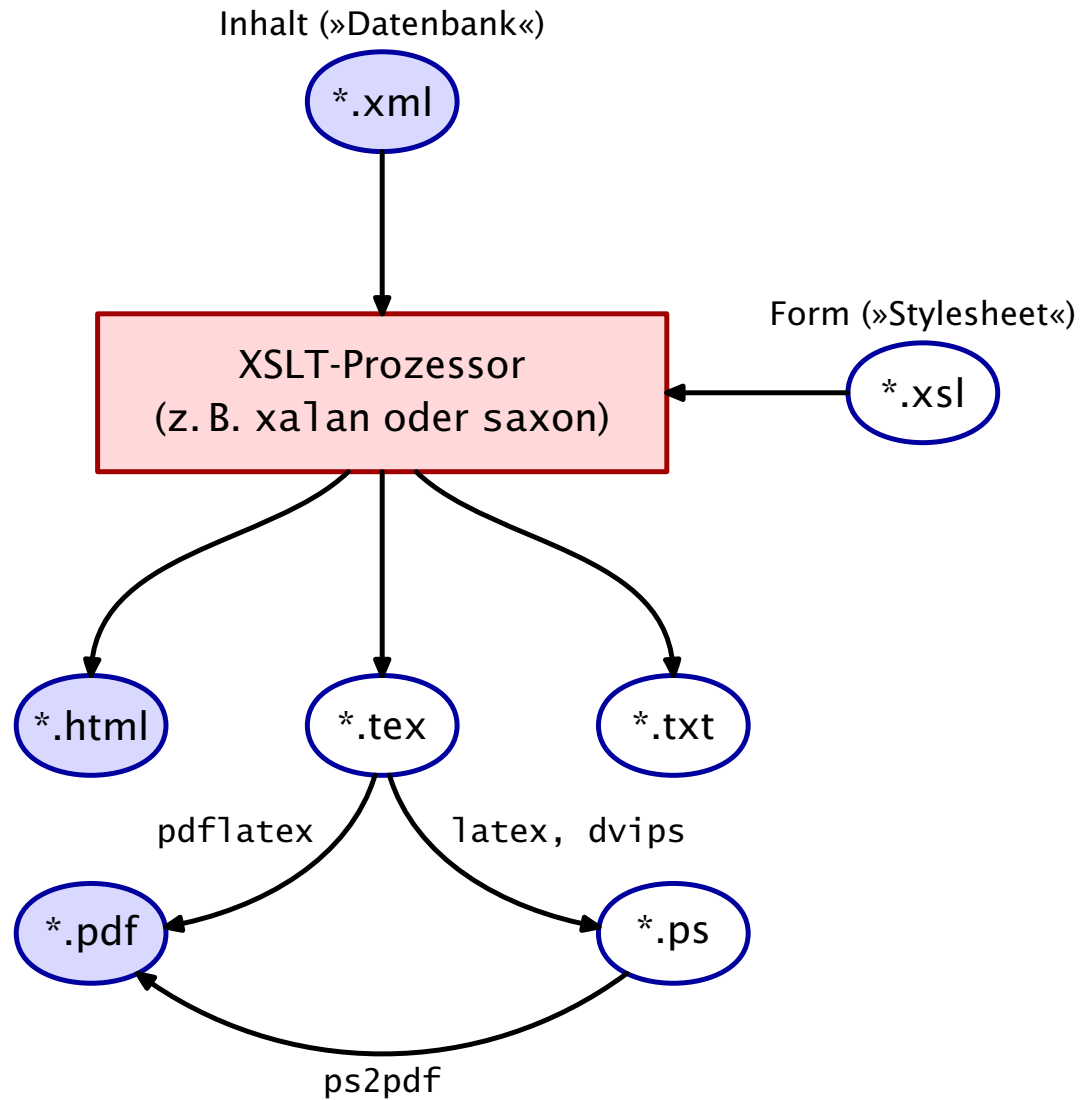
7

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<glossary>
  <entry>
    <term>Rauschgenerator</term>
    <description>
      Getränk, welches bei übermäßigem Genuss
      einen beschwingten Zustand hervorruft.
    </description>
  </entry>
  <entry>
    <term>Chirp</term>
    <description>
      Halluzinatorische Geräuschempfindung
      (z.B. nach dem Genuss von <see>Rauschgeneratoren</see>)
    </description>
  </entry>
</glossary>
```

Vorteile von XML zur Datenerfassung

- weltweit offener und einsehbarer Standard
- unabhängig von Rechnerplattform und Anwendung
- firmenspezifische Software bestimmt nicht, was lesbar/verarbeitbar ist, jeder kann Tools zur XML-Bearbeitung beitragen
- Plain-Text-Format (menschenslesbar, evtl. beschädigte Dateien können „von Hand“ repariert werden, die Inhalte sind immer zugänglich)
- Text auch nach Jahren noch verarbeitbar

Auswahl möglicher Verarbeitungsschritte



Beispiel für ein XSLT-Stylesheet: Serienbrief

10

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!-- Beginn des Stylesheets -->
```

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<!-- Einstellungen bezüglich der Ausgabeart -->
```

```
<xsl:output method="text"
            indent="no"
            media-type="text/plain"
            encoding="ISO-8859-1"
            omit-xml-declaration="yes" />
```

```
<xsl:strip-space elements="*" />
```

Beispiel für ein XSLT-Stylesheet: Serienbrief

11

```
<!-- Vorlage für das Wurzelement: Erzeuge Dokumentenrumpf -->
<xsl:template match="/">\documentclass[fontsize=11pt,
    paper=a4,DIV=12,locfield=wide,hfssiart,
    ]{scr1ttr2}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[ngerman]{babel}
\begin{document}
\setkomavar{title}{Anmeldebestätigung}

    <!-- Ein Brief an jeden Teilnehmer, der noch keinen erhalten hat -->
    <xsl:apply-templates
        select="seminar/teilnehmer[@bestaetigt = 'nein' and @jahr = '2005']" >
        <xsl:sort select="nachname" order="ascending" data-type="text" />
    </xsl:apply-templates>

\end{document}
</xsl:template>
```

Beispiel für ein XSLT-Stylesheet: Serienbrief

12

```
<!-- Vorlage für die Ausgabe eines Briefes -->
<xsl:template match="teilnehmer" >
<xsl:choose> <!-- Fallunterscheidung: Geschlechtsspezifische Anrede -->
  <xsl:when test="anrede = 'Frau'">
\begin{letter}{Frau \\
  <xsl:value-of select="grad" /><xsl:text> </xsl:text>
  <xsl:value-of select="vorname" /><xsl:text> </xsl:text>
  <xsl:value-of select="nachname" /> \\
  <xsl:value-of select="schule" /> \\
  <xsl:value-of select="strasse" /> \\[\medskipamount]
  \textbf{<xsl:value-of select="plz" /><xsl:text> </xsl:text>
  <xsl:value-of select="ort" />}}
\opening{Sehr geehrte Frau <xsl:value-of select="nachname" />,>}
</xsl:when>
```

Beispiel für ein XSLT-Stylesheet: Serienbrief

13

```
<xsl:when test="anrede = 'Herr'">
\begin{letter}{Herrn \\
    <xsl:value-of select="grad" /><xsl:text> </xsl:text>
    <xsl:value-of select="vorname" /><xsl:text> </xsl:text>
    <xsl:value-of select="nachname" /> \\
    <xsl:value-of select="schule" /> \\
    <xsl:value-of select="strasse" /> \\[\medskipamount]
    \textbf{<xsl:value-of select="plz" /><xsl:text> </xsl:text>
    <xsl:value-of select="ort" />}}
\opening{Sehr geehrter Herr <xsl:value-of select="nachname" /> ,}
</xsl:when>
</xsl:choose>      <!-- Ende der Fallunterscheidung -->
wir bestätigen den Eingang Ihrer Anmeldung. Das Seminar findet wie
angekündigt statt und Sie sind als Teilnehmer vorgesehen.

\closing{Mit freundlichen Grüßen}
\end{letter}
</xsl:template>    <!-- Ende des Templates -->
</xsl:stylesheet> <!-- Ende des Stylesheets -->
```

<code>xsl:apply-imports</code>	<code>xsl:import</code>	<code>xsl:text</code>
<code>xsl:apply-templates</code>	<code>xsl:key</code>	<code>xsl:transform</code>
<code>xsl:attribute</code>	<code>xsl:message</code>	<code>xsl:value-of</code>
<code>xsl:attribute-set</code>	<code>xsl:namespace-alias</code>	<code>xsl:variable</code>
<code>xsl:call-template</code>	<code>xsl:number</code>	<code>xsl:when</code>
<code>xsl:choose</code>	<code>xsl:otherwise</code>	<code>xsl:with-param</code>
<code>xsl:comment</code>	<code>xsl:output</code>	
<code>xsl:copy</code>	<code>xsl:param</code>	
<code>xsl:copy-of</code>	<code>xsl:preserve-space</code>	
<code>xsl:decimal-format</code>	<code>xsl:processing-instruction</code>	
<code>xsl:element</code>	<code>xsl:sort</code>	
<code>xsl:fallback</code>	<code>xsl:strip-space</code>	
<code>xsl:for-each</code>	<code>xsl:stylesheet</code>	
<code>xsl:if</code>	<code>xsl:template</code>	

XPath-Funktionen

boolean	last	system-property
ceiling	local-name	translate
concat	name	true
contains	namespace-uri	unparsed-entity-uri
count	normalize-space	
current	not	
document	number	
element-available	position	
false	round	
floor	starts-with	
format-number	string	
function-available	string-length	
generate-id	substring	
id	substring-after	
key	substring-before	
lang	sum	

Literatur

- [1] Stefan Münz: *SELFHTML*. <http://aktuell.de.selfhtml.org/>
- [2] Stefan Münz: *Professionelle Websites*. München : Addison-Wesley, 2006
- [3] Michael Niedermair: *XML für Print und Screen*. Poing : Franzis, 2002
- [4] Saxon: <http://saxon.sourceforge.net/>
- [5] Xalan: <http://xalan.apache.org/>