Verwaltungsaufgaben lösen mit XML und LaTeX

Uwe Siart
TeX-Stammtisch München
tutorien@siart.de

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

Beispiele für Verwaltungsaufgaben

Listen

- Adressenlisten
- Teilnehmerlisten
- Namensschilder
- Tagungsprogramme
- Ressourcenübersichten

Serienbriefe

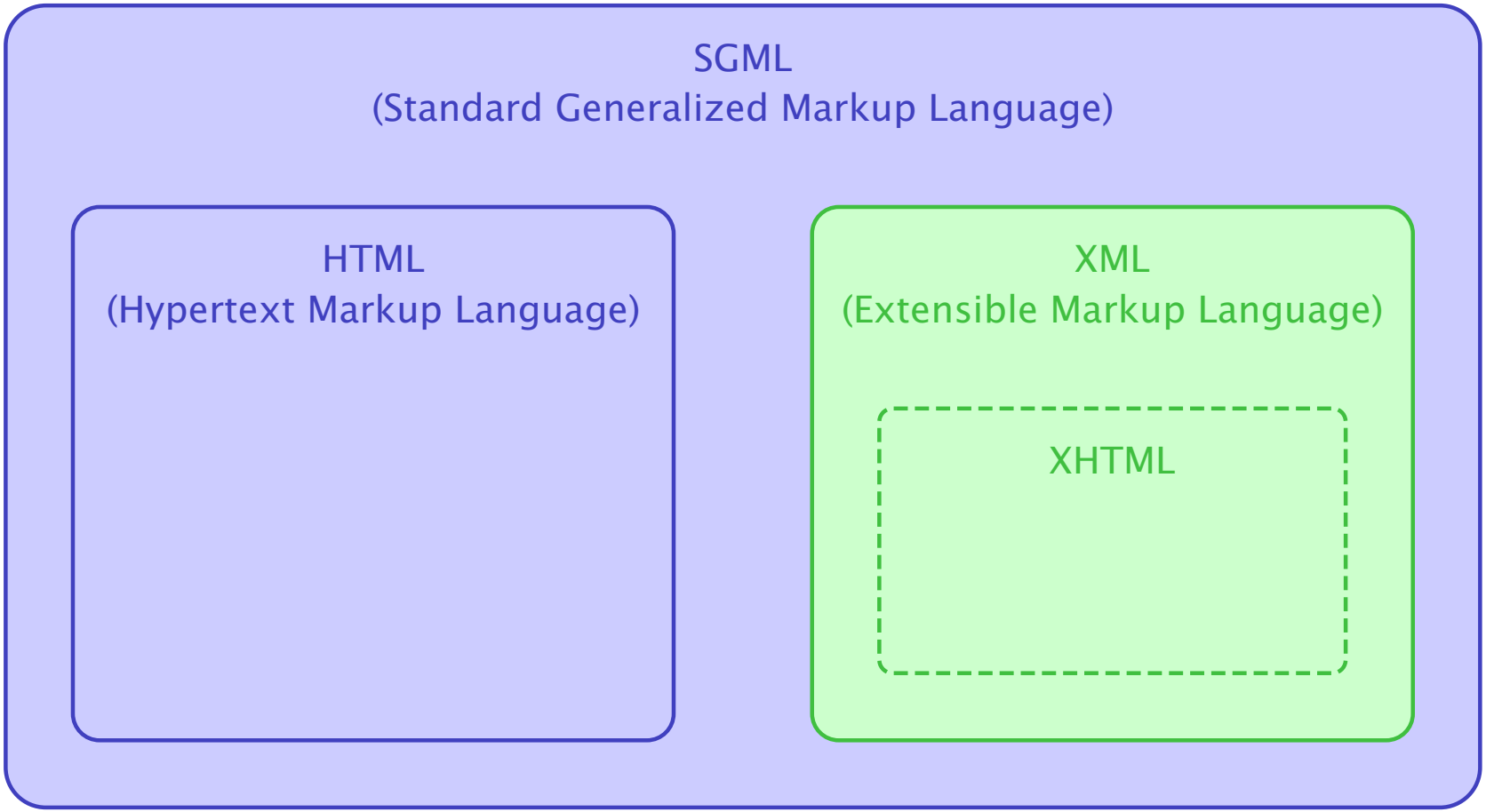
- Anmeldebestätigungen
- Rechnungen

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.

Eigenschaften der Sprache XML

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-Hoppla-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>
```

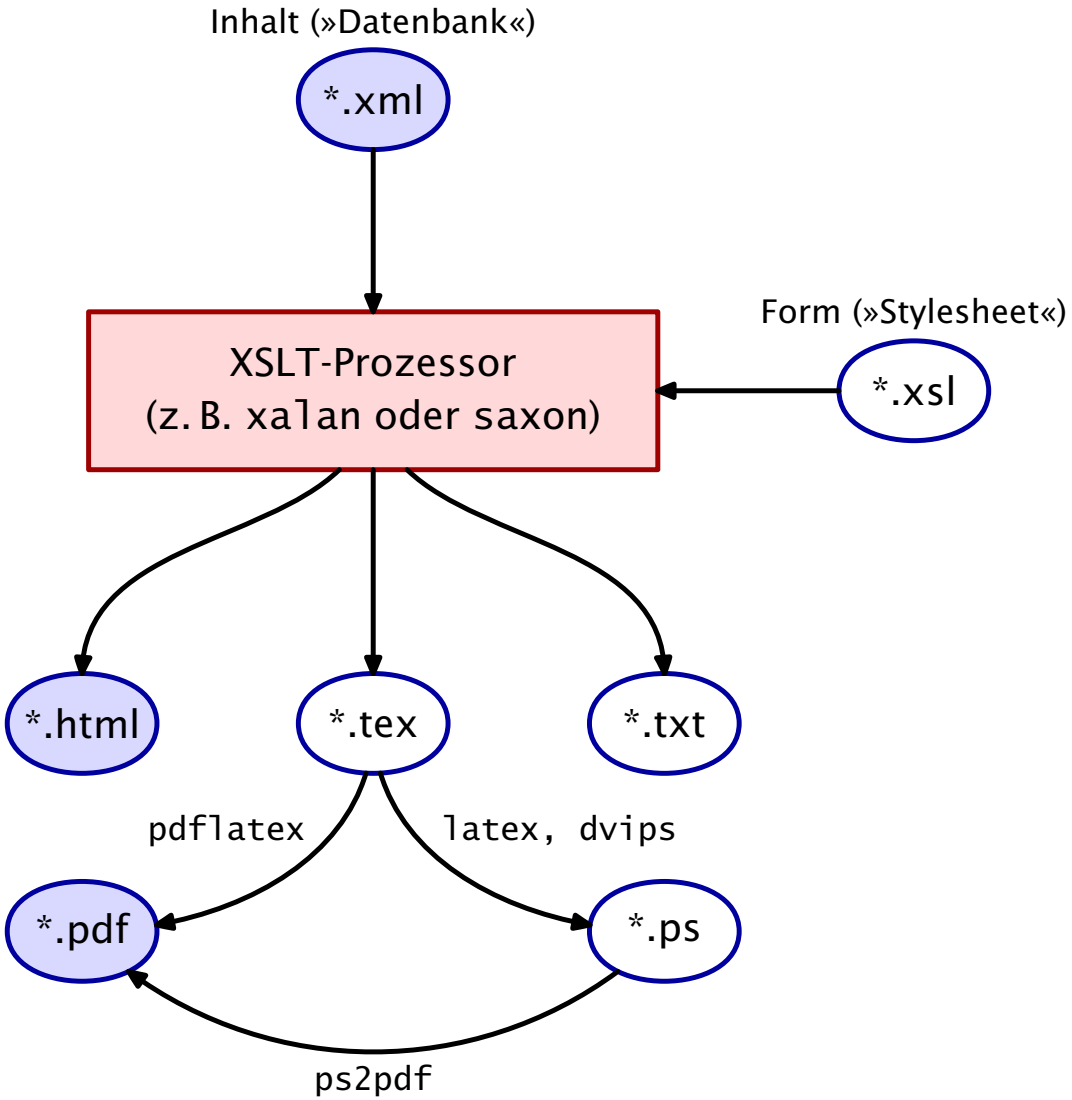
Beispiel für ein XML-Dokument


```
<?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 (menschenlesbar, 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


```
<?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


```
<!-- Vorlage für das Wurzelelement: Erzeuge Dokumentenrumpf -->
<xsl:template match="/">\documentclass[fontsize=11pt,
                paper=a4,DIV=12,locfield=wide,hfssiart,
               ]{scrlttr2}
\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


```
<!-- 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


```
  <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 -->
```

XSLT-Elemente


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

XPath-Funktionen


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

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/