



<http://www.therealgang.de/>

Titel :	Einführung in XML (Teil 2/2)
Author :	Dr. Pascal Rheinert
Kategorie :	Programmierung-Sonstige

Teil II: XML (Dokument 2/2)

Wichtiger Hinweis: Das folgende Dokument basiert auf dem HTML-Lehrgang „Selfhtml“ in der Version 8.0 von Stefan Münz, auf den an sehr vielen Stellen verwiesen wird.

V. Darstellung von XML-Daten (xml\darstellung\)

- **XML-Darstellung mit CSS Stylesheets (css.htm):**
 - Aufbau und Anwendung von CSS für HTML → bereits behandelt
 - Voraussetzung zur Anwendung für XML: erfüllt ab IE 5.0; Netscape 6.0
 - Zusammenhängendes Beispiel: Zugfahrplan → selfhtml
 - Vorteil gegenüber „reinem“ HTML: Die Daten haben eine Aussagekraft und können weiterverarbeitet werden.
 - Vorteil gegenüber „reinem“ XML: Die Daten können im Browser ansprechend dargestellt werden.
 - Bemerkungen zu dem Beispiel:
 - Sehr einfache Möglichkeit, XML grafisch darzustellen. (CSS-Styles tragen dieselben Namen wie die XML-Elemente).
 - Verwendung von „class“ nur aus Darstellungszwecken sollte unterbleiben (→ Frage: Verbergen sich dahinter relevante Dateien? Versteht jeder XML-Parser dies?)
 - Nachteile gegenüber XSL(T):
 - Kombination XML/CSS wird erst von neueren Browsern verstanden.
 - Es lassen sich keine Attribute definieren sondern nur Elemente.
 - Eingeschränkte Funktionsvielfalt (keine automatische Nummerierung bei Elementen etc.)
- **Grundlagen von XSL/XSLT (xslgrundlagen.htm):**
 - XSL (eXtensible Stylesheet Language) besteht aus zwei Komponenten:
 - Formatierungs-Komponente: XSL-FO (XSL-Formatting Objects)
= Stylesheet-Sprache für XML
 - Transformations-Komponente: XSLT (XSL-Transformation)
 - XSL enthält wesentlich mehr Komponenten als CSS (z.B. automatische Nummerierung, bedingte Formatierung etc.).
 - XSLT erlaubt die „Umwandlung“ von XML-Daten in HTML-Daten zum Zweck der Darstellung im Browser.
 - Erfolgt durch ein „Mapping“ von XML-Elementen auf HTML-Tags
Beispiel: Aus <name>Meyer</name> wird Meyer
 - Anschauliche Zuordnung der Elemente der jeweiligen Sprache über „Quellbaum“ (XML) und Ergebnisbaum (HTML) → selfhtml
 - Umwandlung erfolgt durch einen Parser.
 - Dieser kann (sinnvollerweise) serverseitig integriert werden (im Webserver) oder clientseitig.
 - Nachteil clientseitig: Browser muss diese Umwandlung durchführen können (Parser muss integriert sein: erst bei neueren Browsern der Fall und auch dort fehlerhaft)

- **Beispiele für XSLT (xsltbeispiele.htm):**

- Beispiel für serverseitigen Parser: Programm „saxon“
Aufruf: saxon meinedaten.xml meine.xsl >ausgabe.html
- Beispiel 1: Gruss → selfhtml
Benötigt werden 3 Dateien: gruss.dtd; gruss.xml; gruss.xsl

```
gruss.dtd:
<!ELEMENT gruss (#PCDATA)>
-----
gruss.xml:
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE gruss SYSTEM "gruss.dtd">
<?xml-stylesheet type="text/xsl" href="gruss.xsl" ?>

<gruss>hallo Welt!</gruss>
```

```
Gruss.xsl:
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html><head></head><body>
  <p align="center" style="font-family:Tahoma; font-size:48pt;
color:red">
  <xsl:value-of select="." />
  </p>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```

- Gruss.dtd → unverändert
- Gruss.xml → Referenz auf gruss.xsl
(<?xml-stylesheet type="text/xsl" href="gruss.xsl">)
- Gruss.xsl:
 - Es handelt sich um eine XML-artige Datei → 1. Zeile wie bei XML-Nutzdaten-Datei
 - <xsl:...> ... </xsl:...> Notation für XSL-Elemente.
 - Wichtig:
<xsl:template match="xml-element"> </xsl:template>
ordnet dem Element *xml-element* die dort definierte Formatierung zu.
(,/" steht für root)
 - Wichtig:
<xsl:value-of select="." />
fügt den kompletten Inhalt (zwischen <xml-element> und </xml-element> ein
- Zweites Beispiel: „Text mit Formatierungen“ → selfhtml
 - Wichtig:
<xsl:apply-templates /> → „Wende die Formatierungen an, die für die weiteren
(Unter-)Elemente noch definiert werden.

- Drittes Beispiel: „Text mit Hyperlinks (Grafik und Multimedia)“ → selfhtml
 - Umwandlung von `<link>test</link>` in `test` durch:

```
<xsl:template match="link">
<a><xsl:attribute name="href"><xsl:value-of select="." /></xsl:attribute>
<xsl:value-of select="." /></a>
</xsl:template>
```

- Viertes Beispiel: „Glossar“ → selfhtml
 - Wichtig:

```
<xsl:for-each select="glossar/eintrag"> ... </xsl:for-each>
```

bearbeitet alle Elemente

```
<glossar>
  <eintrag>...</eintrag>
  <eintrag>...</eintrag>
  ....
</glossar>
```
 - Wichtig:

Aus diesen Elementen werden über `<xsl:value-of select="beschreibung">` wird das (Unter-)Element `<beschreibung>...</beschreibung>` gewählt und gemäß der Definition formatiert.
 - Wichtig:

Zugriff auf ein Attribut eines Elements erfolgt über `<xsl:value-of select="@attributname">`.

- **XSLT-Elemente (xsltelemente.htm):**
 - () `xsl:apply-imports` (importierte Stylesheets anwenden)
 - Ermöglicht modularen Aufbau der xsl-Dateien.
 - `xsl:apply-templates` (untergeordnete Schablonen anwenden)
 - `xsl:attribute` (Attribut im Ergebnisbaum erzeugen) → Beispiel!
 - () `xsl:attribute-set` (Mehrere Attribute im Ergebnisbaum erzeugen)
 - Definiert eine Art „Attribut-Vorlage“
 - () `xsl:call-template` (Schablone aufrufen)
 - `xsl:choose` (Auswahl treffen)
 - Vergleichbar mit „select case wert“
 - `xsl:comment` (Kommentar setzen)
 - () `xsl:copy` (Element in Ergebnisbaum kopieren)
 - () `xsl:copy-of` (Beliebige ermittelte Daten in Ergebnisbaum kopieren)
 - () `xsl:decimal-format` (Dezimalformat)
 - () `xsl:element` (Element erzeugen)
 - () `xsl:fallback` (alternative Ausführungsanweisung)
 - `xsl:for-each` (für jedes Element aus einer Menge wiederholen)
 - `xsl:if` (wenn-Bedingung)
 - `xsl:import` (Stylesheets importieren)
 - () `xsl:key` (Schlüssel für Auswahl von Elementen)
 - () `xsl:message` (Meldung beim Transformieren ausgeben)

Dr. Pascal Rheinert
Vorlesung „HTML / XML“ – Wintersemester 2001/2002
Akademie der Saarwirtschaft

- () `xsl:namespace-alias` (Stylesheet in anderes Stylesheet transformieren)
- `xsl:number` (fortlaufende Nummerierung)
- `xsl:otherwise` (andernfalls-Anweisung)
- () `xsl:output` (Erzeugen des Ergebnisbaums kontrollieren)
- () `xsl:param` (Parameter definieren)
- () `xsl:preserve-space` (Leerraumzeichen beibehalten)
- () `xsl:processing-instruction` (Stylesheet-Code generieren)
- `xsl:sort` (Elemente nach Inhalt sortieren)
- () `xsl:strip-space` (Behandlung von Leerraumzeichen steuern)
- `xsl:stylesheet` (Stylesheet-Wurzelement)
- `xsl:template` (Schablone für Ergebnisbaum definieren)
- `xsl:text` (Zeicheninhalt ausgeben)
- () `xsl:transform` (Stylesheet-Wurzelement)
- `xsl:value-of` (Wert ausgeben)
- `xsl:variable` (Variable definieren)
- `xsl:when` (Bedingung innerhalb einer Auswahl)
- () `xsl:with-param` (Parameter einen Wert zuweisen)