Referat XML/XSLT

- XML
- XSLT
- Beispiele

Jens Herrmann
Gruppe 18

XML

- Extensible Markup Language
- Ist eine Auszeichnungssprache
- Erweitert von der Standard Generalized Markup Language
- Es gibt viele Einsatzmöglichkeiten

XML

- XML ist frei definierbar, aber...
- ...Wohlgeformtheit beachten:
  - Ein Wurzelelement in der Struktur
  - Alle Elemente besitzen Start- und Ende-Tags
  - Ein Element darf nicht mehrere Attribute mit dem selben Namen besitzen
- ...Validität beachten:
  - Es muss eine Dokumenttypdefinition (DTD) oder
  - Ein XML-Schema, dass etwas moderner ist im Gegensatz zu einer DTD, die keine Namensräume kennt, nicht zwischen Texten und Zahlen unterscheidet und dazu in einer eigenen Sprache verfasst werden muss. Ein XML-Schema ist ein eigenes XML-Dokument

XML

- Die logische Struktur einer XML-Datei ist wie folgt:
  - Elemente: Start-Tag (<Tag-Name>) und End-Tag (</Tag-Name>)
  - Attribute: (Attribut-Name="Attribut-Wert") → Metainformationen
  - Verarbeitungsinformationen: <?Ziel-Name Parameter ?>
  - Kommentare: <!-- Kommentar-Text -->

XML

- Die Darstellung solcher Dateien kann direkt im Browser (Firefox, etc) durch eingebaute Parser, die diese Dateien verarbeiten können
- Per Document Object Model (DOM) können XML-Dokumente eingelesen und wahlfrei bearbeitet werden → sehr speicherintensiv

XML

- Es existieren viele Anwendungsmöglichkeiten für XML:
  - XHTML
  - SVG
  - GPS Exchange Format
  - SMIL
  - RDF
  - OWL
  - Sowie als Export aus vielen Büroanwendungen oder in Programmiersprachen eingebunden

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://localhost:8080/OWLBuergerInformation.owl#"
  xml:base="http://localhost:8080/OWLBuergerInformation.owl">

  <owl:Ontology rdf:about=""/>

  <owl:Class rdf:ID="Gender"/>
  <owl:Class rdf:ID="Person"/>
  <owl:Class rdf:ID="Woman">
    <rdfs:subClassOf rdf:resource="#Person"/>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#Gender"/>
        <owl:hasValue rdf:resource="#female" rdf:type="#Gender"/>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="gender"
     rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
    <rdfs:range rdf:resource="#Gender"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="name"
     rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="firstname"
     rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:DatatypeProperty>

  <Person rdf:ID="STilgner" firstname="Susanne" name="Tilgner">
    <Gender rdf:resource="#female"/>
  </Person>
</rdf:RDF>
```

```
<?xml version="1.0" encoding="UTF-8” ?>
<!DOCTYPE Adressbuch SYSTEM “adress.dtd”>
<Adressbuch>
<Person>
<Name>
<Nachname>Mustermann</Nachname>
<Vorname>Seneca</Vorname>
</Name>
<Adresse>
<Strasse>Gasse</Strasse>
<Hausnummer>42</Hausnummer>
<PLZ>101010</PLZ>
<Ort>Binärstadt</Ort>
</Adresse>
<EMail>11001100@101101.10</EMail>
</Person>
</Adressbuch>
```

XSLT

- Extensible Stylesheet Language Transformation
- Ist eine turing-vollständige Programmiersprache
- Die Stylesheets basieren selbst auf XML
- XSLT erzeugt aus XML-Derivaten dabei Transformationen in der XML-Syntax. Das resultierende Dokument kann dabei auch eines in XML sein, oder aber auch eine Textdatei oder eine Binärdatei.

XSLT

- Template Rules
  - Werden angewendet, sobald eine bestimmte Bedingung passt
- Conflict Rules
  - Sollten mehrere Bedingungen zutreffen, müssen Ausnahmeregeln nach Priorität berücksichtigt werden:
    - Importierte Regeln haben geringere Priorität
    - Ist ein Attribut „priority“ wird dieses berücksichtigt
    - Spezifizierte Pattern haben eine höhere Priorität
    - Gleiche Regeln erzeugen einen Feher

XSLT

- Eine Anwendung ist Presentation Oriented Publishing (POP)
  - Meint eine Transformation zur Darstellung
- Message Oriented Middleware
  - Meint eine Transformation zum Datenaustausch

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

   <xsl:output
      method="xml"
      doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
      doctype-public="-//W3C//DTD XHTML 1.1//EN"
   />

   <xsl:template match="html:body">
      <xsl:copy>
         <xsl:apply-templates select="@*"/>
         <h1><xsl:apply-templates select="//html:title//text()" /></h1>
         <h2 id="toc">Inhaltsverzeichnis</h2>
         <ul>
            <li><a href="#toc">Inhaltsverzeichnis</a></li>
            <xsl:for-each select=".//html:h2">
               <li>
                  <xsl:if test="not(@id)">
                     <xsl:message>Achtung: Kann ohne Id keinen Link erzeugen</xsl:message>
                  </xsl:if>
                  <a href="#{@id}"><xsl:apply-templates/></a>
               </li>
            </xsl:for-each>
         </ul>
         <xsl:apply-templates/>
      </xsl:copy>
   </xsl:template>

   <xsl:template match="node()|@*">
      <xsl:copy>
         <xsl:apply-templates select="node()|@*"/>
      </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

Quellen

- http://de.wikipedia.org/wiki/Xslt
- http://www.xml-xslt.de/index.php?id=xslt
- http://de.wikipedia.org/wiki/Extensible_Markup_Language