Kapitel 10

XML

Die *eXtensible Markup Language (XML)* ist ein vom World-Wide-Web-Konsortium vorgeschlagener Dokumentenverarbeitungsstandard, veröffentlicht unter `http://www.w3c.org/XML` . XML ermöglicht dem Benutzer die Strukturierung seiner Daten mit Hilfe von selbstgewählten *Tags*:

```
<Titel>Selber Atmen</Titel>
<Autor>Willi Wacker</Autor>
```

Mit Hilfe der *eXtensible Stylesheet Language (XSL)* (`http://www.w3c.rog/style/xsl`) kann für die einzelnen Tags eine individuelle Darstellungsweise festgelegt werden, welche auf die zugehörigen Daten angewendet wird.

Auf diese Weise wird eine Trennung zwischen Struktur, Inhalt und Layout erreicht. Typischerweise verteilen sich daher die Angaben zu den Benutzerdaten, z.B. einen Zeitschriftenartikel, auf drei Dateien:

- `artikel.dtd`: *Document Type Definition* mit der Strukturbeschreibung
- `artikel.xml`: *XML-Datei* mit den durch *Tags* markierten Daten
- `artikel.xsl`: *Stylesheet* mit Angaben zum Rendern des Layout

Ein XML-Parser kann zu einer vorliegenden XML-Datei ohne Angabe der zugehörigen DTD überprüfen, ob die XML-Datei *wohlgeformt* ist, d.h. ob die grundsätzlichen Syntaxregeln eingehalten werden. Bei Vorlage der DTD kann der XML-Parser zusätzlich überprüfen, ob die Datei *gültig* ist, d.h. ob ihr Inhalt der Strukturbeschreibung gehorcht. Ein XSLT-Prozessor (*eXtensible Stylesheet Language Transformation)* rendert das Layout für die Daten der XML-Datei unter Anwendung des Stylesheets.

Die Beispiele der folgenden Seiten lassen sich im Microsoft Internet Explorer ausführen nach Installation zweier Dateien aus dem Archiv msxml3.zip . Nach dem Download installiert man zunächst den Microsoft XML-Parser `msxml3.exe` und startet danach die Datei `xmlInst.exe`, welche durch einen Eintrag in der Registry den bisherigen Parser durch den neu installierten ersetzt.

Unter der Adresse `http://xml.apache.org/xerces-j/index.html` gibt es den Public-Domain-Parser *Apache Xerces* zum Download.

Unter der Adresse `http://www.stg.brown.edu/service/xmlvalid` wird eine Validierung von XML-Dokumenten durch Angabe ihrer URL angeboten.

10.1 Strukturierte Texte

Listing 10.1 zeigt die Document Type Definition für die typische Struktur eines Zeitschriftenartikels. Listing 10.2 zeigt den Inhalt des Dokuments, welches diese Strukturvorgaben erfüllt. Abbildung 10.1 zeigt die von Listing 10.3 definierte Ausgabe.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT artikel      (Kopf,Rumpf)>
<!ELEMENT Kopf         (Titel,Autor)>
<!ELEMENT Rumpf        (Kapitel*)>
<!ELEMENT Kapitel      (Titel, Absatz*)>
<!ELEMENT Titel        (#PCDATA)>
<!ELEMENT Autor        (#PCDATA)>
<!ELEMENT Absatz       (#PCDATA|kursiv)*>
<!ELEMENT kursiv       (#PCDATA)>
```

Listing 10.1: artikel.dtd

```
<?xml version='1.0' encoding="ISO-8859-1" ?>
<!DOCTYPE artikel SYSTEM "artikel.dtd" >
<artikel>
  <Kopf>
    <Titel>Selber Atmen</Titel>
    <Autor>Willi Wacker</Autor>
  </Kopf>
  <Rumpf>
    <Kapitel>
      <Titel>Einleitung</Titel>
      <Absatz>
        In einer Reihe von aufsehenerregenden Experimenten
        wurden krzlich am <kursiv>Max-Planck-Institut</kursiv>
        die Vorteile des selbstndigen Denkens herausgearbeitet.
      </Absatz>
      <Absatz>
        Unsere Forschungen lassen erwarten, da analoge
        Aussagen auch fr den Atmungsvorgang gelten knnten.
      </Absatz>
    </Kapitel>
    <Kapitel>
      <Titel>Ausblick</Titel>
      <Absatz>
        Es gibt viel zu tun; warten wir es ab !
      </Absatz>
    </Kapitel>
  </Rumpf>
</artikel>
```

Listing 10.2: artikel.xml

```
<?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="artikel">
    <html>
      <head><title><xsl:value-of select="Titel"/></title></head>
      <body><xsl:apply-templates/></body>
    </html>
  </xsl:template>

  <xsl:template match="artikel/Kopf/Titel">
    <center><h1><xsl:apply-templates/></h1></center>
  </xsl:template>

  <xsl:template match="artikel/Kopf/Autor">
    <center><h4><xsl:apply-templates/></h4></center>
  </xsl:template>

  <xsl:template match="artikel/Rumpf/Kapitel/Titel">
    <h3><xsl:apply-templates/></h3>
  </xsl:template>

  <xsl:template match="Absatz">
    <p><xsl:apply-templates/></p>
  </xsl:template>

  <xsl:template match="kursiv">
    <I><xsl:apply-templates/></I>
  </xsl:template>

</xsl:stylesheet>
```

Listing 10.3: artikel.xsl

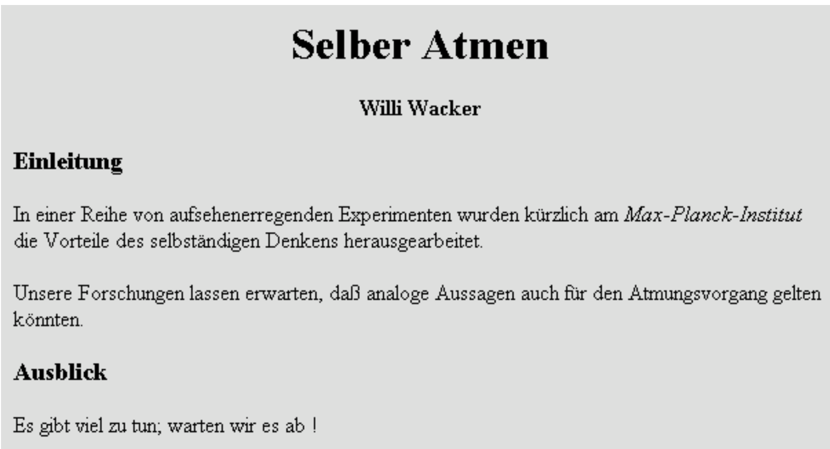


Abbildung 10.1: Ausgabe von artikel.xml mit Hilfe von artikel.xsl

10.2 Strukturierte Daten

Listing 10.4 zeigt die Document Type Definition für einen Entity-Typ *personen*, bestehend aus den Feldern *Vorname, Nachname* und *Photo* und den Attributen *PersNr, Chef* und *Geschlecht*. Das Feld *Photo* ist leer und besteht nur aus einem Attribut *src* zur Angabe einer Bilddatei. Das Attribut *PersNr* vom Typ `ID` muß eindeutig sein, das Attribut *Chef* bezieht sich auf ein ID-Attribut. Listing 10.5 zeigt einige zur DTD passende Daten.

```
<?xml version='1.0' encoding="ISO-8859-1" ?>
<!ELEMENT personen              (person*)>
<!ELEMENT person                (vorname,nachname,photo)>
<!ATTLIST person   persnr      ID #REQUIRED>
<!ATTLIST person   chef        IDREF #IMPLIED>
<!ATTLIST person   geschlecht  (mnnlich|weiblich) #REQUIRED>
<!ELEMENT vorname               (#PCDATA)>
<!ELEMENT nachname              (#PCDATA)>
<!ELEMENT photo                  EMPTY>
<!ATTLIST photo    src           CDATA #REQUIRED>
```

Listing 10.4: personen.dtd

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE personen SYSTEM "personen.dtd">
<personen>
  <person persnr="P4711" chef="P4712" geschlecht="weiblich" >
    <vorname>Erika</vorname>
    <nachname>Mustermann</nachname>
    <photo src="erika.gif"/>
  </person>
  <person persnr="P4712" geschlecht="mnnlich">
    <vorname>Willi</vorname>
    <nachname>Wacker</nachname>
    <photo src="willi.gif"/>
  </person>
</personen>
```

Listing 10.5: personen.xml

Listing 10.6 zeigt eine Document Type Definition für den Entity-Typ *Dozenten*, der den Entity-Typ *Professoren* erweitert um eine variable Anzahl von Feldern mit dem Namen *Amt*. Abbildung 10.2 zeigt Teil einer Ausprägung des XML-Baums. Listing 10.7 zeigt dazu passende Daten.

```
<?xml version='1.0' encoding="ISO-8859-1" ?>
<!ELEMENT dozenten    (dozent*)>
<!ELEMENT dozent      (PersNr,Name,Rang,Raum,Amt*)>
<!ELEMENT PersNr      (#PCDATA)>
<!ELEMENT Name        (#PCDATA)>
<!ELEMENT Rang        (#PCDATA)>
<!ELEMENT Raum        (#PCDATA)>
<!ELEMENT Amt         (Bezeichnung,Termin?)>
<!ELEMENT Bezeichnung (#PCDATA)>
<!ELEMENT Termin      (#PCDATA)>
```

Listing 10.6: dozenten.dtd

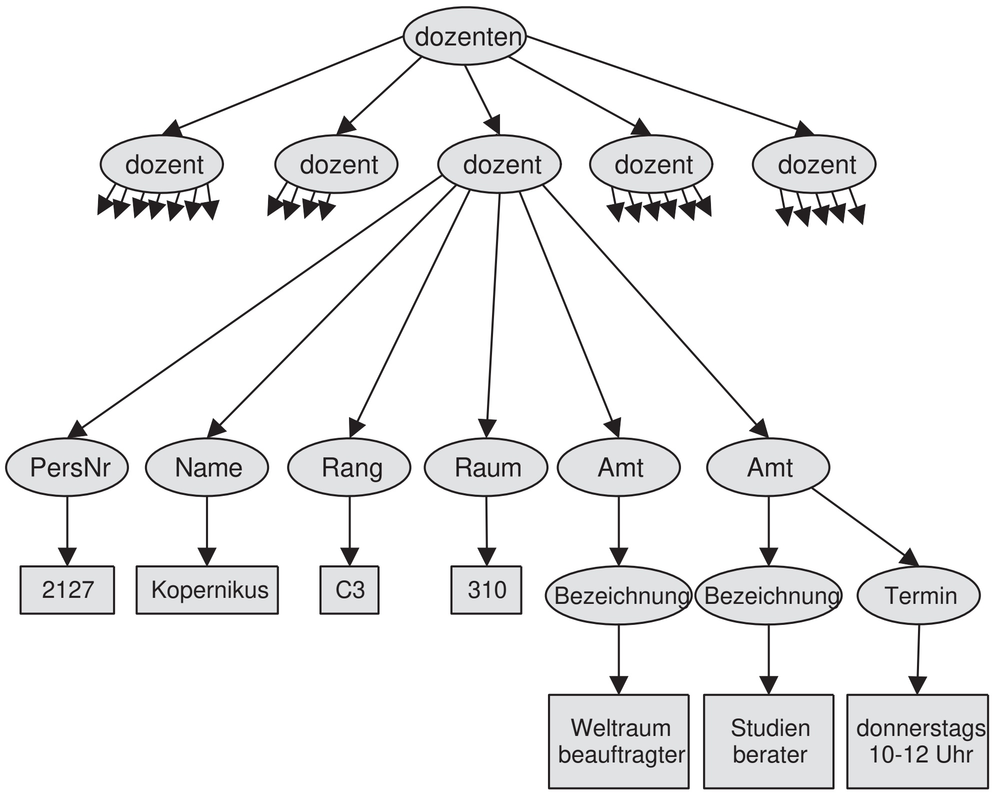


Abbildung 10.2: XML-Baum zur Datei dozenten.xml

```xml
<?xml version='1.0' encoding="ISO-8859-1" ?>
<!DOCTYPE dozenten SYSTEM "dozenten.dtd" >
<dozenten>
  <dozent>
    <PersNr>2125</PersNr>
    <Name>Sokrates</Name>
    <Rang>C4</Rang>
    <Raum>226</Raum>
    <Amt>
      <Bezeichnung>Dekan</Bezeichnung>
      <Termin>vormittags</Termin>
    </Amt>
  </dozent>
  <dozent>
    <PersNr>2126</PersNr>
    <Name>Russel</Name>
    <Rang>C4</Rang>
    <Raum>232</Raum>
  </dozent>
  <dozent>
    <PersNr>2127</PersNr>
    <Name>Kopernikus</Name>
    <Rang>C3</Rang>
    <Raum>310</Raum>
    <Amt>
      <Bezeichnung>Weltraumbeauftrager</Bezeichnung>
    </Amt>
    <Amt>
      <Bezeichnung>Studienberater</Bezeichnung>
      <Termin>donnerstags 10-12 Uhr</Termin>
    </Amt>
  </dozent>
  <dozent>
    <PersNr>2133</PersNr>
    <Name>Popper</Name>
    <Rang>C3</Rang>
    <Raum>52</Raum>
  </dozent>
  <dozent>
    <PersNr>2134</PersNr>
    <Name>Augustinus</Name>
    <Rang>C3</Rang>
    <Raum>309</Raum>
  </dozent>
</dozenten>
```

Listing 10.7: dozenten.xml

10.3 XML-Ausgabe auf Web-Seiten

Alle Beispiele dieser Sektion (mit Ausnahme des letzten) beziehen sich auf die Document Type Definition `dozenten.dtd`, gezeigt in Listing 10.6, sowie die XML-Datei `dozenten.xml`, gezeigt in Listing 10.7.

Listing 10.8 zeigt den Quelltext einer HTML-Datei `dozenten-tabelle.html`, in der die Elemente der XML-Datei `dozenten.xml` zu einer Tabelle aufbereitet werden. Abbildung 10.3 zeigt die Ausgabe.

```
<html>
  <head>
    <title>dozenten-tabelle.html</title>
  </head>
  <body bgcolor="DDDDDD">
    <xml id="dozenten" src="dozenten.xml"></xml>
    <table border="1" bgcolor="ABCDEF" datasrc="#dozenten">
      <tr>
        <td><span datafld="Name"></span></td>
        <td><span datafld="Rang"></span></td>
      </tr>
    </table>
  </body>
</html>
```

Listing 10.8: Ausgabe einer XML-Datei als HTML-Tabelle

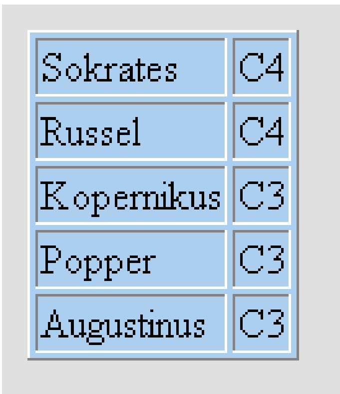


Abbildung 10.3: Ausgabe einer XML-Datei als HTML-Tabelle

Listing 10.9 zeigt den Quelltext der HTML-Datei `dozenten-traverse.html`, in der mittels VB-Script der Microsoft XML-Parser verwendet wird, um die Knoten des XML-Baumes der XML-Datei `dozenten.xml` zu traversieren. Abbildung 10.4 zeigt die Ausgabe.

```
<html>
  <head>
    <title>dozenten-traverse.html</title>
  </head>
  <body bgcolor="dddddd">
    <script type="text/vbscript">
      set xmlDoc=CreateObject("Microsoft.XMLDOM")
      xmlDoc.async="false"
      xmlDoc.load("dozenten.xml")
      document.write("<h1>Traversieren der XML Knoten</h1>")
      for each x in xmlDoc.documentElement.childNodes
        document.write("<b>" & x.nodename & "</b>")
        document.write(": ")
        document.write(x.text)
        document.write("<br><br>")
      next
    </script>
  </body>
</html>
```

Listing 10.9: Traversieren der XML-Knoten

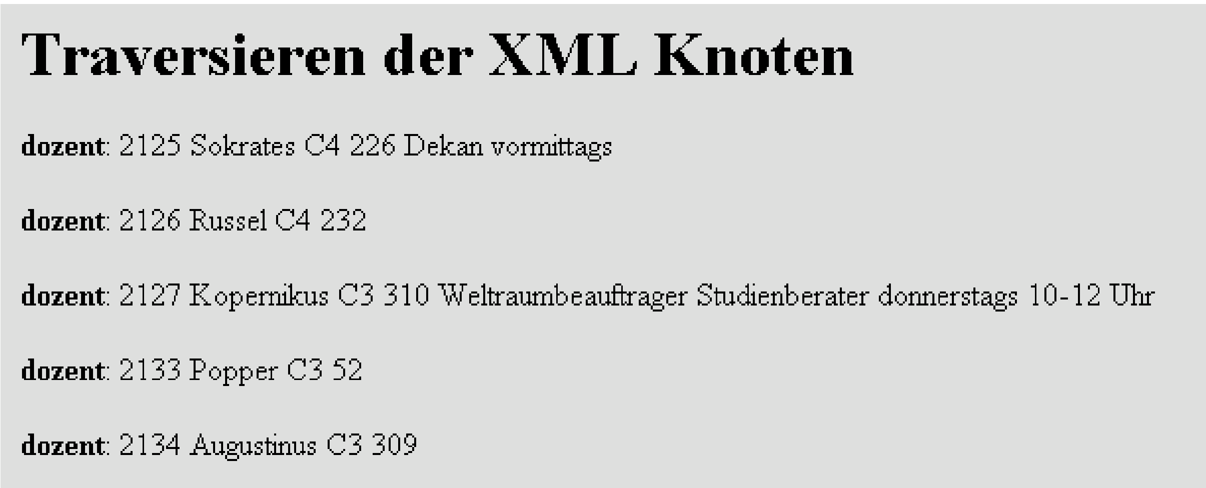


Abbildung 10.4: Traversieren der XML-Knoten

Listing 10.10 zeigt den Quelltext der HTML-Datei `dozenten-navigation.html`, in der mit Hilfe von Javascript durch die Elemente der XML-Datei `dozenten.xml` navigiert wird. Abbildung 10.5 zeigt die Ausgabe.

```
<html>
  <head><title>dozenten-navigation.html</title></head>
    <script type="text/javascript">
      function naechste(){
        x=vor.recordset
        if (x.absoluteposition < x.recordcount){ x.movenext()}
      }
      function vorige(){
        x=vor.recordset
        if (x.absoluteposition > 1){ x.moveprevious()}
      }
    </script>
  </head>
  <body bgcolor="dddddd">
    <xml src="dozenten.xml" id="vor" async="false"></xml>
    <B>Name:</B> <span datasrc="#vor" datafld="Name"></span><br>
    <B>Rang:</B> <span datasrc="#vor" datafld="Rang"></span><br>
    <B>Raum:</B> <span datasrc="#vor" datafld="Raum"></span><br>
    <input type="button" value="Voriger Dozent"  onclick="vorige()">
    <input type="button" value="Nchster Dozent" onclick="naechste()">
  </body>
</html>
```

Listing 10.10: Navigation durch eine XML-Datei mit Hilfe von Javascript

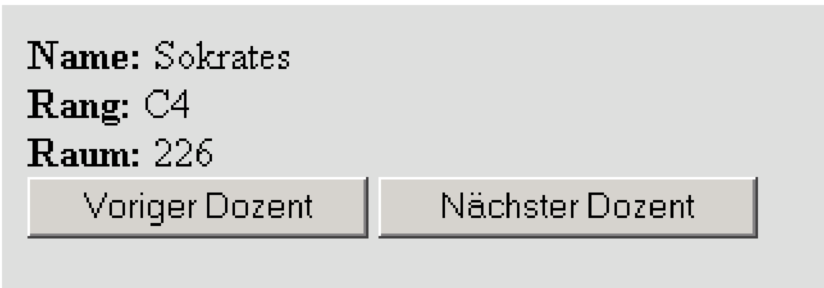


Abbildung 10.5: Navigation durch die XML-Datei mit Hilfe von Javascript

Listing 10.11 zeigt den Quelltext der Cascading-Stylesheet-Datei `dozenten.css`, auf die von der XML-Datei `dozenten.xml` Bezug genommen werden kann, nachdem dort die Zeile

```
<?xml-stylesheet type="text/css" href="dozenten.css"?>
```

eingefügt worden ist. Abbildung 10.6 zeigt die Ausgabe.

```
dozenten{background-color: #dddddd;width: 100%;}
dozent {display: block;   margin-bottom: 10pt;margin-left: 15pt;}
PersNr {color:    #0000FF; font-size: 20pt;}
Name   {color:    #FF0000; font-size: 28pt;}
Rang   {color:    #0000FF; font-size: 20pt;}
Raum   {color:    #0000FF; font-size: 20pt;}
Amt    {Display: block;   color: #000000;margin-left: 20pt;}
```

Listing 10.11: Cascading Stylesheet für dozenten.xml

Abbildung 10.6: Ausgabe von dozenten.xml mit Hilfe von dozenten.css

Listing 10.12 zeigt den Quelltext der XSL-Datei `dozenten.xsl`, auf die von der XML-Datei `dozenten.xml` Bezug genommen werden kann über die Zeile

```
<?xml-stylesheet type="text/xsl" href="dozenten.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>
    <body>
      <table border="1" bgcolor="ABCDEF">
        <tr>
          <th>Pers-Nr</th><th>Name</th><th>Rang</th><th>Raum</th><th>mter</th>
        </tr>
        <xsl:for-each select="/dozenten/dozent">
          <tr>
            <td><xsl:value-of select="PersNr" /></td>
            <td><xsl:value-of select="Name"   /></td>
            <td><xsl:value-of select="Rang"   /></td>
            <td><xsl:value-of select="Raum"   /></td>
            <td>
               <xsl:for-each select="Amt">
                 <xsl:value-of select="Bezeichnung" />
                 <xsl:if test="position() != last()">, </xsl:if>
               </xsl:for-each>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Listing 10.12: dozenten.xsl

Pers-Nr	Name	Rang	Raum	Ämter
2125	Sokrates	C4	226	Dekan
2126	Russel	C4	232	
2127	Kopernikus	C3	310	Weltraumbeauftrager, Studienberater
2133	Popper	C3	52	
2134	Augustinus	C3	309	

Abbildung 10.7: Ausgabe von dozenten.xml mit Hilfe von dozenten.xsl

Eine Alternative zur Anwendung einer XSL-Datei auf eine XML-Datei besteht in der Formulierung einer HTML-Seite, in der mit Hilfe des Microsoft XML-Parsers zunächst die XML-Datei und die XSL-Datei geladen werden und danach die Transformation angestoßen wird. Listing 10.13 zeigt den Quelltext der HTML-Datei `dozenten-mit-xsl.html`.

In beiden Fällen kommt dieselbe XSL-Datei zur Anwendung, das Ergebnis wird in Abbildung 10.7 gezeigt.

```
<html>
  <head><title>dozenten-mit-xsl.html</title></head>
  <body bgcolor="dddddd">
    <script type="text/javascript">

      var xml = new ActiveXObject("Microsoft.XMLDOM") // XML-File laden
      xml.async = false
      xml.load("dozenten.xml")

      var xsl = new ActiveXObject("Microsoft.XMLDOM") // XSL-File laden
      xsl.async = false
      xsl.load("dozenten.xsl")

      document.write(xml.transformNode(xsl))          // transformieren

    </script>
  </body>
</html>
```

Listing 10.13: HTML-Datei zur Ausgabe von dozenten.xml mit Hilfe von dozenten.xsl

Das folgende Beispiel bezieht sich auf die im Listing 10.4 gezeigte Document Type Definition `personen.dtd`. Neben dem Attribut *persnr* vom Typ `ID` zur eindeutigen Kennung einer Person und dem Attribut *chef* vom Tpy `IDREF` zur Referenz auf eine persnr gibt es ein leeres Feld *Photo* mit dem Attribut *src* zur Angabe einer Bilddatei.

Listing 10.14 zeigt den Quelltext der XSL-Datei `personen.xsl`, auf die von der XML-Datei `personen.xml` Bezug genommen werden kann, nachdem die Zeile

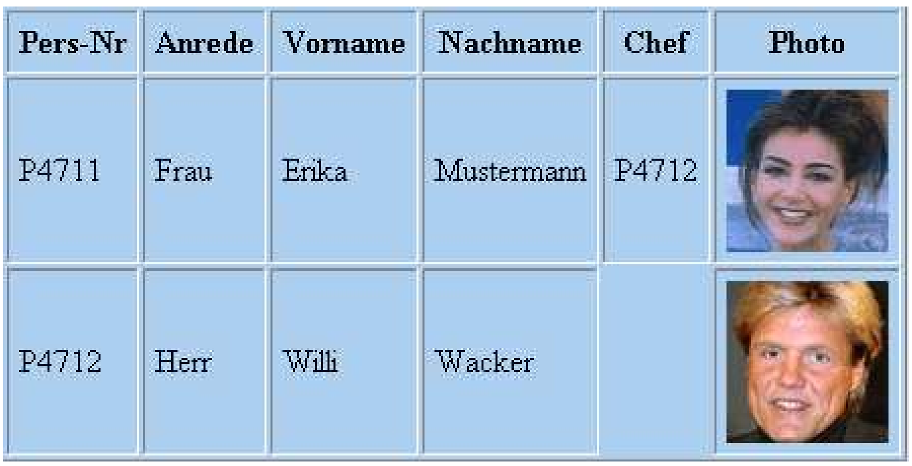
```
<?xml-stylesheet type="text/xsl" href="personen.xsl"?>
```

eingefügt wurde. Abbildung 10.8 zeigt die Ausgabe.

```
<?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>
    <body bgcolor="dddddd">
      <table border="1" cellpadding="5" bgcolor="abcdef">
        <tr>
          <th>Pers-Nr</th><th>Anrede</th><th>Vorname</th><th>Nachname</th>
          <th>Chef</th><th>Photo</th>
        </tr>
        <xsl:for-each select="/personen/person">
          <tr>
            <td><xsl:value-of select="@persnr"/></td>
            <td>
             <xsl:if test="@geschlecht=’weiblich’">Frau</xsl:if>
             <xsl:if test="@geschlecht=’mnnlich’">Herr</xsl:if>
            </td>
            <td><xsl:value-of select="vorname"    /></td>
            <td><xsl:value-of select="nachname"   /></td>
            <td><xsl:value-of select="@chef"      /></td>
            <td><IMG SRC="{photo/@src}"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Listing 10.14: personen.xsl



Pers-Nr	Anrede	Vorname	Nachname	Chef	Photo
P4711	Frau	Erika	Mustermann	P4712	
P4712	Herr	Willi	Wacker		

Abbildung 10.8: Ausgabe von personen.xml mit Hilfe von personen.xsl

10.4 XML vom Microsoft SQL Server 2000

Der Microsoft SQL-Server 2000 unterstützt XML sowohl bei der Überführung von XML-Dokumenten in die Datenbank als auch bei der Wiedergabe von Tabellendaten in XML-Notation.

Durch Aufruf der stored procedure `sp_xml_prepare_document` kann der wohlgeformte Inhalt eines XML-Strings in eine interne Darstellung überführt werden. Durch Aufruf von `OPENXML` kann der Inhalt des XML-Baums als Rowset geliefert werden. Listing 10.15 zeigt den gesamten Aufruf, Abbildung 10.9 zeigt die erzeugte Tabelle.

```
DECLARE @hdoc int                         -- Handle fuer interne Darstellung
DECLARE @doc varchar(1000)                -- Variable fuer XML-String
SET @doc =                                -- initialisiere XML-String
'<dozenten>
    <dozent>
      <persnr>4711</persnr>
      <name>Willi</name>
      <rang>C4</rang>
      <raum>801</raum>
    </dozent>
 </dozenten>'

EXEC sp_xml_preparedocument               -- lege interne Dartstellung @hdoc
     @hdoc OUTPUT, @doc                   -- fuer den XML-String @doc an

SELECT * FROM OPENXML                     -- werte XML-Baum aus
       (@hdoc,'/dozenten/dozent')         -- ueber Handle @hdoc

EXEC sp_xml_removedocument @hDoc          -- entferne interne Darstellung
```

Listing 10.15: Aufruf der Prozedur sp_xml_prepare_document

	id	parentid	nodetype	localname	prefix	namespaceuri	datatype	prev	text
1	2	0	1	dozent	NULL	NULL	NULL	NULL	NULL
2	3	2	1	persnr	NULL	NULL	NULL	NULL	NULL
3	7	3	3	#text	NULL	NULL	NULL	NULL	4711
4	4	2	1	name	NULL	NULL	NULL	3	NULL
5	8	4	3	#text	NULL	NULL	NULL	NULL	Willi
6	5	2	1	rang	NULL	NULL	NULL	4	NULL
7	9	5	3	#text	NULL	NULL	NULL	NULL	C4
8	6	2	1	raum	NULL	NULL	NULL	5	NULL
9	10	6	3	#text	NULL	NULL	NULL	NULL	801

Abbildung 10.9: XML-Baum als Tabelle, erstellt von OPENXML

Der Inhalt der von XMLOPEN gelieferten Tabelle kann mit Hilfe eines `INSERT`-Statements in eine bestehende Datenbanktabelle eingefügt werden. Hierbei müssen über das Schlüsselwort `WITH` die zu verwendenden Spaltentypen angegeben werden. Dem OPENXML-Kommando wird durch die Flagge

`1` eine attributbezogene, durch die Flagge `2` eine elementbezogene XML-Struktur signalisiert. Listing 10.16 zeigt den kompletten Ablauf.

```
DECLARE @hdoc int                        -- Handle fuer interne Darstellung
DECLARE @doc varchar(1000)               -- Variable fuer XML-String
SET @doc =                               -- initialisiere XML-String
'<dozenten>
    <dozent>
      <persnr>4711</persnr>
      <name>Willi</name>
      <rang>C4</rang>
      <raum>801</raum>
    </dozent>
    <dozent>
      <persnr>4712</persnr>
      <name>Erika</name>
      <rang>C3</rang>
      <raum>802</raum>
    </dozent>
 </dozenten>'

EXEC sp_xml_preparedocument              -- lege interne Dartstellung @hdoc
     @hdoc OUTPUT, @doc                  -- fuer den XML-String @doc an

insert into professoren                  -- fuege in Tabelle professoren ein
            (persnr,name,rang,raum)      -- Persnr, Name, Rang, Raum
SELECT * FROM OPENXML                    -- werte XML-Baum aus
       (@hdoc,'/dozenten/dozent',2)      -- Parameter 2: Elementbezogen
with (persnr int, name varchar(20),      -- Datentypen fuer persnr und Name
      rang char(2), raum int)            -- Datentypen fuer Rang und Raum

EXEC sp_xml_removedocument @hDoc         -- entferne interne Darstellung
```

Listing 10.16: Einfügen mit Hilfe von OPENXML

Eine Ausgabe von Tabelleninhalten in geschachtelter XML-Notation ist beim SQL-Server 2000 möglich im Rahmen einer erweiterten SQL-Syntax:

```
select persnr, name from professoren
where rang='C3'
for xml auto, elements
```

Hierbei wird für jede Zeile aus dem Result-Set ein Tupel erzeugt, welches geklammert wird durch Tags mit dem Namen der Tabelle. Jede Spalte des Result-Set wird geklammert durch Tags mit dem Namen der Spalte:

```
<professoren><persnr>2127</persnr><name>Kopernikus</name></professoren>
<professoren><persnr>2134</persnr><name>Augustinus</name></professoren>
```

10.5 XQuery

Als Abfragesprache für XML-Dokumente scheint sich *XQuery* durchzusetzen. Im Frühjahr 2001 wurde unter der Adresse `http://www.w3.org/TR/xquery` ein W3C Working Draft veröffentlicht.

In *XQuery* werden Abfragen durch eine Kombination von *FLWR*-Ausdrücken, Pfadausdrücken und Element-Konstruktoren formuliert. Ein *FLWR*-Ausdruck (gesprochen wie das englische Wort *Flower*) besteht aus einer Sequenz der Schlüsselworte `FOR, LET, WHERE, RETURN`. Ein Pfadausdruck faßt wie die *XML Path Language XPath* (`http://www.w3.org/TR/xpath.html`) das Dokument als Baum von Element-Knoten auf und liefert durch gezielte Navigation den gewünschten Teilbaum ab. Dabei trennt ein einzelner Schrägstrich ('/') Elemente in direkter Vater/Sohn-Beziehung, ein doppelter Schrägstrich ('//') bedeutet eine Nachkommen-Beziehung über eine oder auch mehrere Baum-Ebenen. Ein Element-Konstruktor klammert das Ergebnis der Query mit öffnenden und schließenden Tags.

- Liste alle C4-Dozenten:

```
document("dozenten.xml")/dozenten/dozent[Rang="C4"]
```

- Liste alle Dozenten, die mindestens eine Vorlesung halten:

```
<aktive-Dozenten>
  FOR $d IN document("dozenten.xml")//dozent
  LET $v := document("vorlesungen.xml")//vorlesung[gelesenVon=$d/PersNr]
  WHERE count($v) > 0
  RETURN <Dozent>$d/Name</Dozent>
</aktive-Dozenten>
```

- Liste alle Dozenten zusammen mit ihren Lehrveranstaltungen:

```
<Vorlesungsverzeichnis>
  FOR    $d IN document("dozenten.xml")//dozent,
         $v IN document("vorlesungen.xml")//vorlesung
  WHERE  $d/PersNr = $v/gelesenVon
  RETURN <Veranstaltung>$d/Name, $v/Titel</Veranstaltung>
</Vorlesungsverzeichnis>
```

- Liste zu jedem Dozenten alle seine Lehrveranstaltungen:

```
<Vorlesungsverzeichnis>
  FOR      $d IN document("dozenten.xml")//dozent
  RETURN
    <Dozent>$d/Name</Dozent>
    FOR    $v IN document("vorlesungen.xml")//vorlesung
    WHERE  $d/PersNr = $v/gelesenVon
    RETURN <Vorlesung>$v/Titel</Vorlesung>
</Vorlesungsverzeichnis>
```

- Liste zu jedem Dozenten die Summe der Semesterwochenstunden seiner Lehrveranstaltungen:

```
<Dozenten>
  FOR    $d IN document("dozenten.xml")//dozent
  LET    $s := sum(document("vorlesungen.xml")
                   //vorlesung[gelesenVon=$d/PersNr]/SWS)
  WHERE  $s > 0
  RETURN <Dozent>
           <Name>$d/Name</Name>
           <Lehrbelastung>$s</Lehrbelastung>
         </Dozent>
</Dozenten>
```

- Liste alle Dozenten mit demselben Rang wie Sokrates:

```
<Rang-wie-Sokrates>
  FOR   $d1 IN document("dozenten.xml")//dozent[Name="Sokrates"],
        $d2 IN document("dozenten.xml")//dozent[Rang=$d1/Rang]
  RETURN <Name>$d2/Name</Name>
</Rang-wie-Sokrates>
```