Access & XML

Mark Dörbandt

AEK-Vortrag

2005-10-01 / 2005-10-08

Agenda

- ***Einordnung***
  - *Warum X*?*
  - *XML & Datenbanken*
  - *XML in Access*
- ***Theorie***
  - *XML*
  - *XML-Schema*
  - *XPath*
  - *XSLT*
- ***Praxis***
  - *SysInfo*
  - *Word*
  - *Excel*
  - *HTML*
  - *MySQL*
  - *Google Earth*
- ***Quellen***

Umfrage: wer nutzt XML?

Warum X*?

- **universelle Austauschsprache**
- **Standard (W3C, Branchen)**
- **viel hilft viel:**
  - *Anwendungen*
  - *Funktionen*
  - *Tools*
  - *Bibliotheken*
- **Lesbarkeit**
  - *Struktur und Daten*
  - *Mensch*
  - *Maschine*
- **XML ist Text**
  - *Firewall*
  - *Sicherheit*
  - *Dateigröße*

XML & Datenbanken

- **Strukturiertheit**
  - *Volltext – XML – Datenbank*
- **Dokumente**
- **Hierarchien – Relationen**

- **Datentypen: XML Schema**

- **Datenbanken:**
  - *Massendaten*
  - *Indizierung*
  - *Replikation*
  - *Transaktionen*

XML in Access

- **Access 97**
- **Access XP**
- ***Access 2003*** *Umfrage: wer nutzt Access 2003?*

XML in Access 97

- **keine eingebaute Unterstützung**

- **Nutzung der MSXML-Bibliotheken (DOM)**
  - *Codebeispiel im Skript*

- **Erzeugen und Parsen von Textdateien**
  - *Codebeispiel im Skript*

- **Nutzung von ADO**

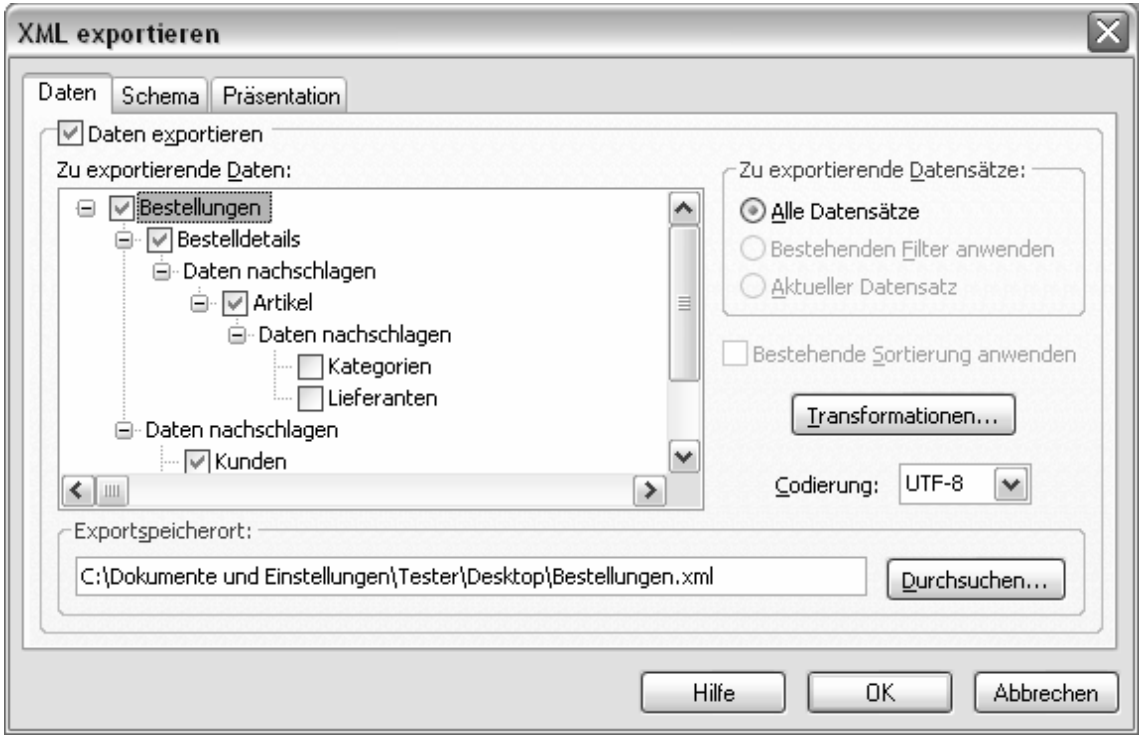
XML in Access XP

- **Import und Export möglich**
  - *XML*
  - *Schema*
  - *Stylesheet / HTML*

- **Import und Export per VBA**
  - *Application.ImportXML*
  - *Application.ExportXML*

XML in Access 2003

- **zwei wesentliche Neuerungen**
  - *Hierarchien (aus Beziehungen)*
  - *Transformationen (XSLT)*



Theorie

- **XML in zehn Minuten**

- **3½*5: XML Survival Package**

Theorie: XML

</>

- **Element**
  <Elementname>Inhalt</Elementname>
  <Elementname/>

- **Attribut**
  <Element Attribut="Wert">

- **Verarbeitungsanweisung**
  <?xml-stylesheet href="my.css" ?>

- **Kommentar**
  <! -- Kommentar -->

- **Namensräume**
  <xsl:template>
  xmlns:xsl="http://...

Theorie: XML-Schema <xs:

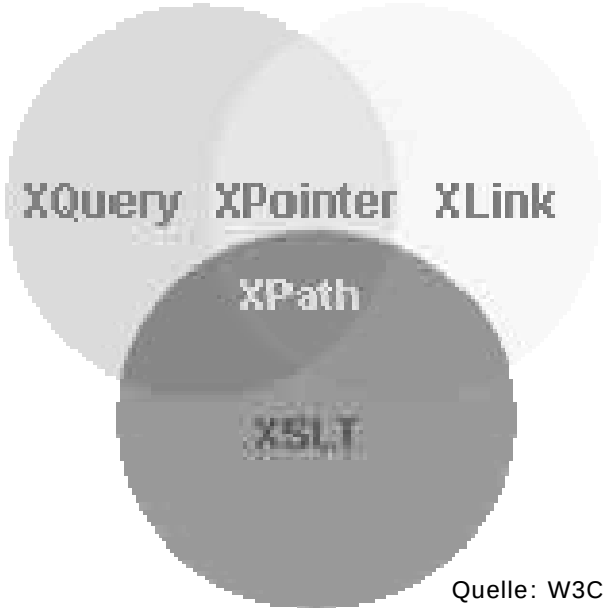
- **Wohlgeformtheit vs. Gültigkeit**

- **Datentypen**
  - *einfache Datentypen*
  - *komplexe Datentypen*
  - *Einschränkung, Erweiterung*
  - *Aufzählung, Muster*

- **Struktur**
  - *Elemente + Attribute*
  - *Hierarchie*
  - *Sequenzen*
  - *Vorkommnisse*

Theorie: XPath ../*

- **Pfade**
  /dataroot/Tabelle/Feld
- **Achsen**
  *./Feld/**
  ../Tabelle
- **Knotentest**
  Tabelle/Feld[Inhalt>5]
- **Funktionen**
  Feld[position() mod 2=0]
- **Prädikat**
  /Feld[@Attribut="Test"]



Quelle: W3C

Theorie: XSLT <xsl:

- **Transformation**
- **Ausgabemedium**
  - *XML*
  - *HTML*
  - *Text*
- **Templates**
  <xsl:template match="/">
- **Funktionen**
  - *position()*
  - *count()*
  - *substring()*
  - *sum()*
  - *format-number()*
- **Bausteine**
  <xsl:if>
  <xsl:for-each>
  <xsl:element>
  <xsl:attribute>
  <xsl:apply-templates>

Beispiele: Überblick

- **Ziele**
  - *Demonstration der neuen Funktionen*
    - manuell
    - automatisiert
  - *Import und Verarbeitung von XML*
  - *Export und Transformation von XML*
    - nach XML (Office: Word, Excel)
    - nach HTML
    - nach Text (SQL)
  - *Analyse des Vorgehens*
    - Code
    - XML
    - Stylesheet

Beispiel: SysInfo (Ziele)

- **Ziel**
  - *Import aus einer XML-Datei*
  - *Automatisierung des Imports*
  - *einfache Transformation, Variable*

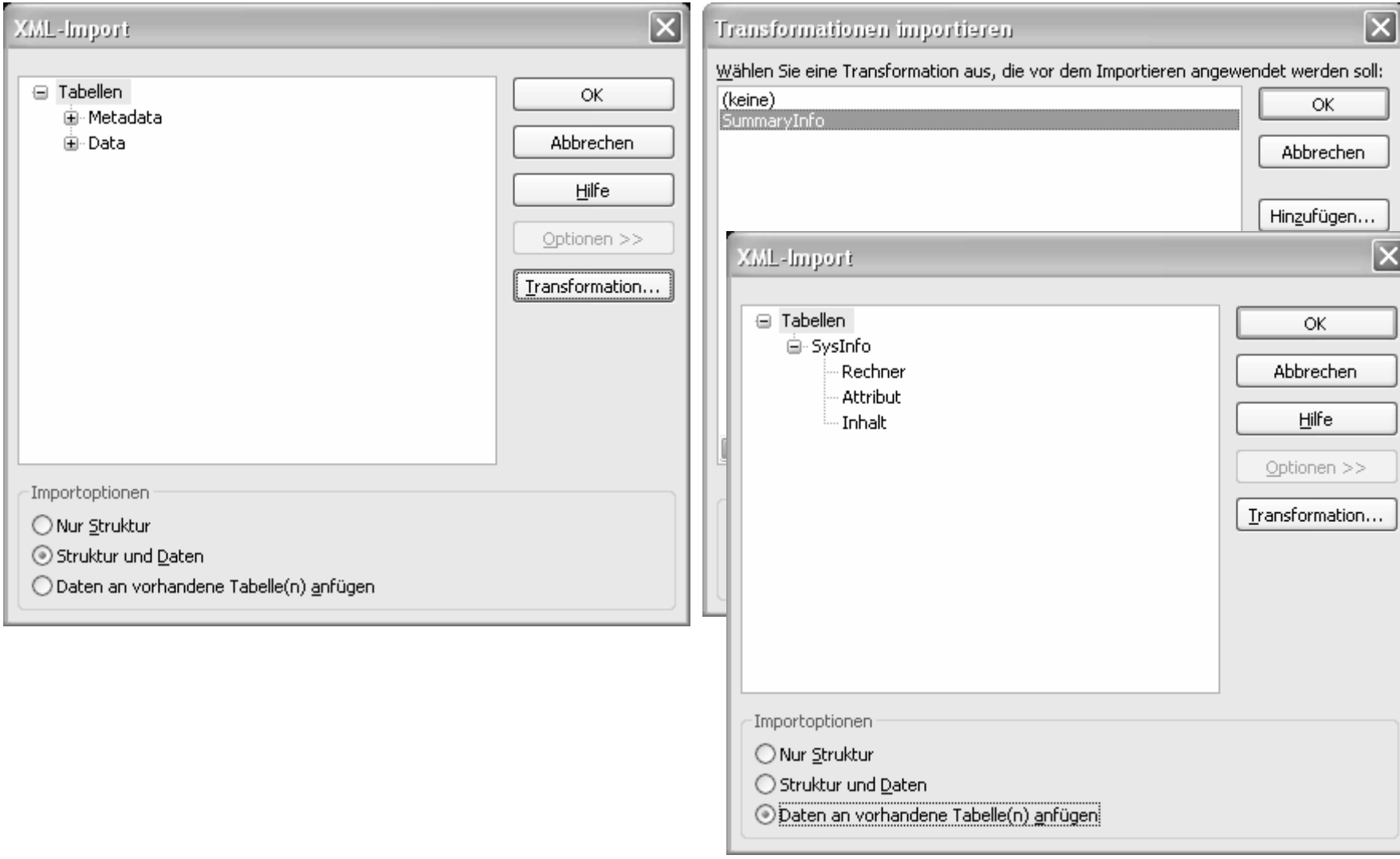
- **Szenario**
  - *Unternehmen mit vielen Rechnern*
  - *Sammeln von System-Informationen*
  - *zentrale Datenbasis*

Beispiel: SysInfo (Demo)

DEMO ...

Beispiel: SysInfo (manuell)

Datei – externe Daten – Importieren – Dateityp: XML



Beispiel: SysInfo (Code)

```
Sub importSysInfo()

  Dim strFilename As String
  strFilename = getFileName("SysInfo", "Import", _
                            "xml", getDBPath)
  If strFilename = "" Then Exit Sub

  Application.TransformXML strFilename, _
                           getDBPath & "\SummaryInfo.XSL", _
                           getDBPath & "\SysInfo.xml"

  Application.ImportXML getDBPath & "\SysInfo.xml", _
                        acAppendData

End Sub
```

Beispiel: SysInfo (XML)

```
<?xml version="1.0"?>
<MsInfo>
  <Metadata>
    <Version>7.0</Version>
    <CreationUTC>09/17/05 12:35:16</CreationUTC>
  </Metadata>
  <Category name="Systemübersicht">
    <Data>
      <Element><![CDATA[Betriebssystemname]]></Element>
      <Wert><![CDATA[Microsoft Windows XP
                Professional]]></Wert>
    </Data>
    <Data>
      <Element><![CDATA[Version]]></Element>
      <Wert><![CDATA[5.1.2600 Service Pack 2
                Build 2600]]></Wert>
    </Data>
    ...
  </Category>
</MsInfo>
```

Beispiel: SysInfo (Stylesheet)

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"  ...>
  <xsl:output method="xml" encoding="UTF-8" indent="yes" />
  <xsl:template match="/">
    <xsl:variable name="System"><xsl:value-of
      select="/MsInfo/Category[@name='Systemübersicht']
      /Data[Element='Systemname']/Wert" /></xsl:variable>
    <xsl:element name="dataroot">
      <xsl:for-each select="MsInfo/Category/Data">
        <xsl:element name="SysInfo">
          <xsl:element name="Rechner">
            <xsl:value-of select="$System"/>
          </xsl:element>
          <xsl:element name="Attribut">
            <xsl:value-of select="Element"/>
          </xsl:element>
          <xsl:element name="Inhalt"><xsl:value-of select="Wert"/>
          </xsl:element>
        </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Beispiel: SysInfo (XML transformiert)

```
<?xml version="1.0"?>
<dataroot>
  <SysInfo>
    <Rechner>DIT8</Rechner>
    <Attribut>OS Name</Attribut>
    <Inhalt>Microsoft Windows XP Professional</Inhalt>
  </SysInfo>
  <SysInfo> ... </SysInfo>
  ...
</dataroot>
```

Beispiel: Word (Ziele)

- **Ziele**
  - *Export in eine XML-Datei (hierarchisch)*
  - *Automatisierung des Exports*
  - *komplexere Transformation*
  - *WordML*

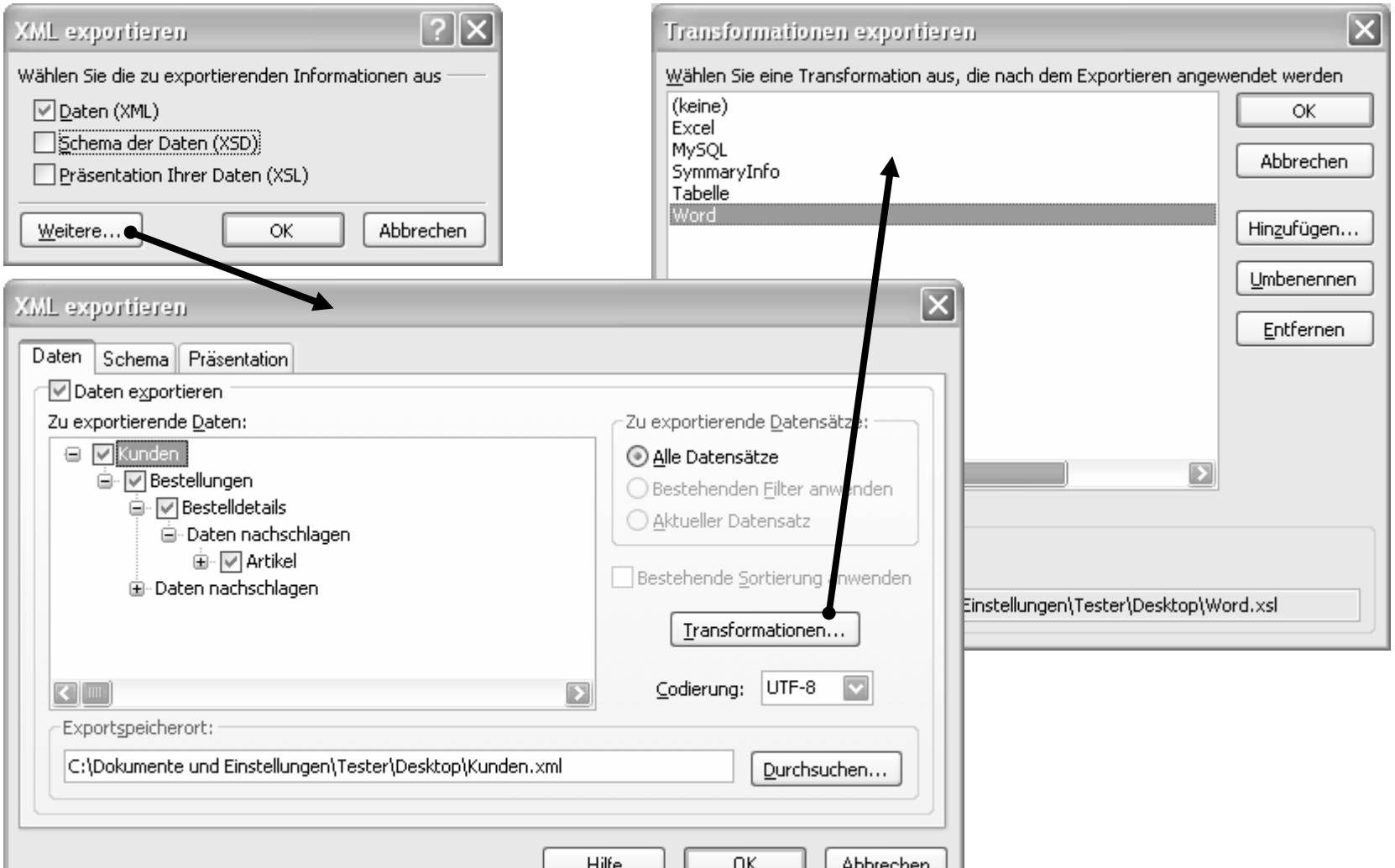
- **Szenario**
  - *NordWind: Bestellungen*
  - *es wird ein Bericht*
    über alle Bestellungen
    als Word-Datei
    benötigt

Beispiel: Word (Demo)

DEMO ...

Beispiel: Word (manuell)

Datei – Exportieren – Dateityp: XML



Beispiel: Word (Code)

```
Sub exportKunden()
  Dim addOrders As AdditionalData
  Dim addOrderDetails As AdditionalData
  Dim addArtikel As AdditionalData
  Dim addDetails As AdditionalData
  Set addOrders = Application.CreateAdditionalData
  Set addOrderDetails = addOrders.Add("Bestellungen")
  Set addArtikel = addOrderDetails.Add("Bestelldetails")
  addArtikel.Add "Artikel"
  Application.ExportXML acExportTable, "Kunden", _
                        getDBPath & "\Kunden.XML", _
                        , , , , , , addOrders
  Application.TransformXML getDBPath & "\Kunden.XML", _
                           getDBPath & "\Word.XSL", _
                           getDBPath & "\Word.XML"
  Application.FollowHyperlink getDBPath & "\Word.XML"
End Sub
```

Beispiel: Word (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot ...>
<Kunden>
  <Kunden-Code>ALFKI</Kunden-Code>
  <Firma>Alfreds Futterkiste</Firma>  ...
  <Bestellungen>
    <Bestell-Nr>10643</Bestell-Nr>
    <Kunden-Code>ALFKI</Kunden-Code>    ...
    <Bestelldetails>
      <Bestell-Nr>10643</Bestell-Nr>
      <Artikel-Nr>28</Artikel-Nr>      ...
    </Bestelldetails>
    <Bestelldetails>...</Bestelldetails>
  </Bestellungen>
  <Bestellungen>...</Bestellungen>
</Kunden>
<Kunden>...</Kunden>
<Artikel><Artikel-Nr>1</Artikel-Nr> ... </Artikel>
<Artikel>...</Artikel>
</dataroot>
```

Beispiel: Word (Stylesheet, Auszug)

```
<xsl:template match="/">
  <xsl:processing-instruction name="mso-application">
  progid="Word.Document"</xsl:processing-instruction> ...
  <xsl:apply-templates select="dataroot" /> ...
</xsl:template>
```

```
<xsl:template match="dataroot">
  Bericht Bestellungen nach Kunden
  <xsl:apply-templates select="Kunden" />
</xsl:template>
```

```
<xsl:template match="Kunden">
  <xsl:value-of select="Firma" /> ...
  <xsl:apply-templates select="Bestellungen" />
</xsl:template>
```

```
<xsl:template match="Bestellungen">
  Bestellung <xsl:value-of select="Bestell-Nr" />
  <xsl:apply-templates select="Bestelldetails" />
</xsl:template>
```

```
<xsl:template match="Bestelldetails">>
  <xsl:value-of select="Anzahl" /> ...
</xsl:template>
```

Beispiel: Excel (Ziele)

- **Ziele**
  - *ähnlich Word*
  - *Excel-XML-Format*
  - *erweiterte Funtionen in Transformationen*

- **Szenario**
  - *es soll eine Trendanalyse für umfangreiche Daten durchgeführt werden*
  - *Export der Daten nach Excel*
  - *Trendfunktion*
  - *Diagramm*

Preisfrage: warum umfangreich?

Beispiel: Excel (Demo)

DEMO ...

Beispiel: Excel (Stylesheet, Ausschnitt)

```
<xsl:variable name="RecCount">
<xsl:value-of select="count(/dataroot/Trend)-1"/></xsl:variable>
<xsl:template match="Trend">
  <xsl:choose>
    <xsl:when test="position()=1">
      <Row>
        <Cell ss:StyleID="s21"><Data ss:Type="DateTime">
          <xsl:value-of select="Datum" /></Data></Cell>
        <Cell><Data ss:Type="Number">
          <xsl:value-of select="Messwert" /></Data></Cell>
        <xsl:element name="Cell">
          <xsl:attribute name="ss:ArrayRange">
            RC:R[<xsl:value-of select="$RecCount" />]C
          </xsl:attribute>
          <xsl:attribute name="ss:Formula">
           =TREND(RC[-1]:R[<xsl:value-of select="$RecCount" />]C[-1]
            RC[-2]:R[<xsl:value-of select="$RecCount" />]C[-2])
          </xsl:attribute>
        </xsl:element>
      </Row>
    </xsl:when>
    <xsl:otherwise>...</xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Beispiel: HTML (Ziele)

- **Ziele**
  - *Export nach HTML*
  - *Darstellung von Kreuztabellen, Sortieren, Gruppieren*

- **Szenario**
  - *in einem Schachverein wird eine Clubmeisterschaft veranstaltet*
  - *die Ergebnisse sollen in einer Datenbank verwaltet und im Internet veröffentlicht werden*

Beispiel: HTML (Demo)

DEMO ...

Beispiel: HTML (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot ...>
  <Tabelle>
    <Spieler>Claus</Spieler>
    <Punkte>2</Punkte>
    <Runde2>1</Runde2>
    <Runde4>1</Runde4>
  </Tabelle>
  <Tabelle>
    <Spieler>Gerd</Spieler>
    <Punkte>1</Punkte>
    <Runde1>1</Runde1>
    <Runde3>0</Runde3>
  </Tabelle>
  ...
</dataroot>
```

Beispiel: HTML (Stylesheet, Auszug)

```
<xsl:key name="Rd"
 match="/dataroot/Tabelle/*[starts-with(name(),'Runde')]„
 use="name()" />
<xsl:template match="Tabelle">
  <xsl:variable name="CurNode" select="." />
  <tr>
    <td align="right"><xsl:value-of select="Spieler" /></td>
    <td align="center"><xsl:value-of select="Punkte" /></td>
    <xsl:for-each
      select="/dataroot/Tabelle/*[starts-with(name(),'Runde')]">
      <xsl:sort select="name()" />
      <xsl:if
       test="generate-id()=generate-id(key('Rd', name())[1])">
        <xsl:variable name="CurName" select="name()" />
        <td align="center">
          <xsl:value-of select="$CurNode/*[name()=$CurName]"/
        </td>
      </xsl:if>
    </xsl:for-each>
  </tr>
</xsl:template>
```

Beispiel: MySQL (Ziele)

- **Ziele**
  - *Export nach Text*
  - *Änderungsprotokoll*
  - *weitere Bausteine in Transformationen, Modi*

- **Szenario**
  - *eine lokale Access.MDB soll im Internet in einer MySQL-Datenbank dupliziert werden*
  - *die lokale Datenbank ist der Master, Änderungen sollen per SQL-Skript remote nachvollzogen werden*

Beispiel: MySQL (Demo)

DEMO ...

Beispiel: MySQL (Code I)

```
Private Sub Form_AfterInsert()
  strKey = Me![Artikel-Nr]
  strNow = Format(Now, "yyyymmddhhnnss") & Format(Rnd * 99, "##")
  CurrentDb.Execute "INSERT INTO Record „ & _
    "(Stamp, Action, TableName, KeyField, KeyValue) " & _
    "VALUES ('" & strNow & "','I','Artikel', _
             '[Artikel-Nr]','" & strKey & "');"
  lngRecordID = Nz(DLookup("ID", "Record", _
                   "Stamp='" & strNow & "'"), -1)
  For Each daoField In Me.Recordset.Fields
    CurrentDb.Execute "INSERT INTO RecordDetail " & _
      "(RecordID,FieldName,OldValue,NewValue,FieldType) " & _
      "VALUES (" & Format(lngRecordID) & ", '" & _
       daoField.Name & "','','" & _
       Format(Me(daoField.Name)) & "'," & daoField.Type & ");
  Next daoField
End Sub

Private Sub Form_Delete(Cancel As Integer)
Private Sub Form_BeforeUpdate(Cancel As Integer)
```

Beispiel: MySQL (Stylesheet, Ausschnitt)

```
<xsl:template match="Record">
  <xsl:choose><xsl:when test="Action='I'">
   <xsl:text>INSERT INTO </xsl:text>
   <xsl:value-of select="TableName"/>
   <xsl:text> (</xsl:text>
   <xsl:apply-templates select="RecordDetail" mode="FieldList"
   <xsl:text>) VALUES (</xsl:text>
   <xsl:apply-templates select="RecordDetail" mode="ValueList"
   <xsl:text>)</xsl:text>
  </xsl:when>
  <xsl:when test="Action='D'"> ... </xsl:when>
  <xsl:when test="Action='U'"> ... </xsl:when>
  </xsl:choose> ...
</xsl:template>
<xsl:template match="RecordDetail" mode="FieldList">
  <xsl:text>[</xsl:text>
  <xsl:value-of select="FieldName"/><xsl:text>]</xsl:text>
  <xsl:if test="position() &lt; count(../RecordDetail)">
    <xsl:text>, </xsl:text></xsl:if>
</xsl:template>
<xsl:template match="RecordDetail" mode="ValueList"> ...
</xsl:template>
```

Beispiel: Google Earth (Ziele)

- **Ziele**
  - *Ermittlung von Geokoordinaten*
  - *Darstellung von Geoinformationen*

- **Szenario**
  - *anhand einer PLZ-Information sollen Geokoordinaten aus einer freien Geo-Datenbank ermittelt und dargestellt werden*

Beispiel: Google Earth (Demo)

DEMO ...

Beispiel: Google Earth (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
  <Placemark>
    <name>MMmK</name>
    <LookAt>
      <longitude>13.39427054019512</longitude>
      <latitude>52.4243713021014</latitude>
      <range>452.7760234923183</range>
      <tilt>70.72567452644698</tilt>
      <heading>9.873351998535547</heading>
    </LookAt>
    <styleUrl>root://styles#default</styleUrl>
    <Point>
      <coordinates>
        13.39247662390909,52.42340263854255,0
      </coordinates>
    </Point>
  </Placemark>
</kml>
```

Beispiel: Google Earth (Stylesheet)

```
<xsl:template match="Point">
  <kml xmlns="http://earth.google.com/kml/2.0">
    <Placemark xmlns="http://earth.google.com/kml/2.0">
      <name><xsl:value-of select="ShowName"/></name>
      <LookAt>
        <longitude><xsl:value-of select="X"/></longitude>
        <latitude><xsl:value-of select="Y"/></latitude>
        <range>452.7760234923183</range>
        <tilt>70.72567452644698</tilt>
        <heading>9.873351998535547</heading>
      </LookAt>
      <styleUrl>root://styles#default</styleUrl>
      <Point>
        <coordinates>
          <xsl:value-of select="X"/>,
          <xsl:value-of select="Y"/>,0
        </coordinates>
       </Point>
    </Placemark>
  </kml>
</xsl:template>
```

Beispiele: Zusammenfassung

- **manueller Import und Export**
- **automatisierte Lösung**
- **Transformationen sehr mächtig, besonderes in Kombination mit Hierarchien**
- **einzelne Aufgaben anspruchsvoll**
- **alle Zielformate möglich**

Quellen

- **Skript**
- **Foliensatz**
- **Beispiele.zip**

- **Webseite** **http://www.doerbandt.de/XML**

- **Bücher**
- **Links**

- **Fragen?**