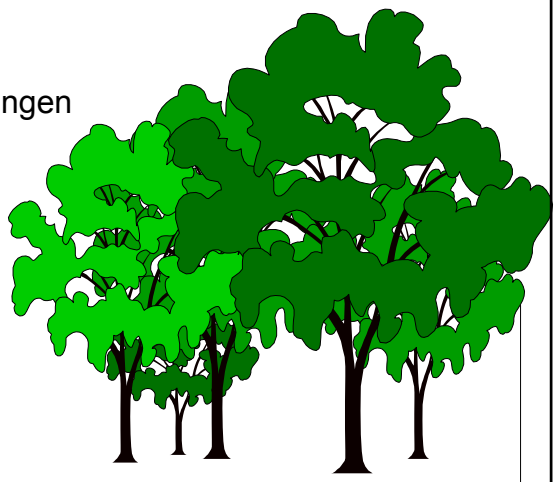
Orientation in Objects

XSLT Muster und Lösungen



Thomas Bayer
Tobias Kieninger
Özgür Kipik

bayer@oio.de

Orientation in Objects GmbH
Weinheimer Str. 68
68309 Mannheim
www.oio.de

Inhalt

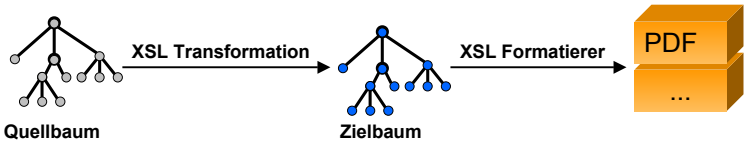
Orientation in Objects

- Einleitung
  - XSL & XSLT
  - XPath
  - XML Pattern
- XSLT Patterns
  - XSLT Stylesheet Patterns
  - XSLT Patterns
- XSLT Praxis Techniken und Beispiele
  - Mengenoperationen
  - Techniken zu Sortieren, Gruppieren,...
  - Inhaltsverzeichnis erstellen
  - Cross References
  - Best Practise

XSL

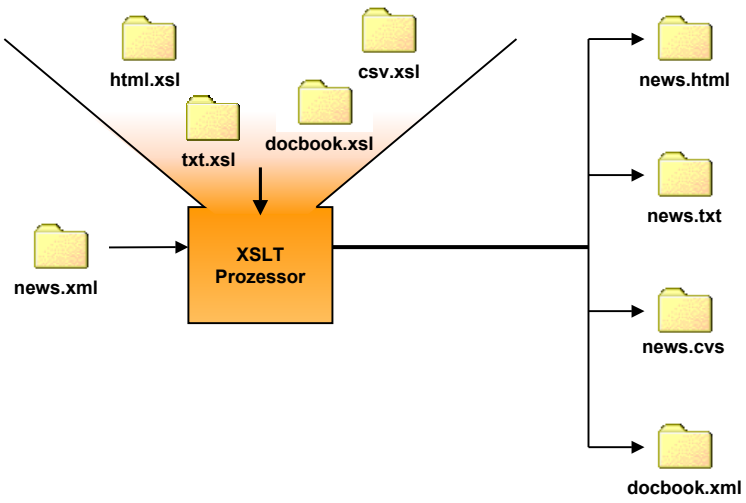
Orientation in Objects

- Besteht aus zwei Teilen:
  - Der Transformationssprache XSLT
    - Ändern der Struktur
    - Hinzufügen von Formatierungsanweisungen
  - XML Vokabular für Formatierungen
    - Darstellungsform

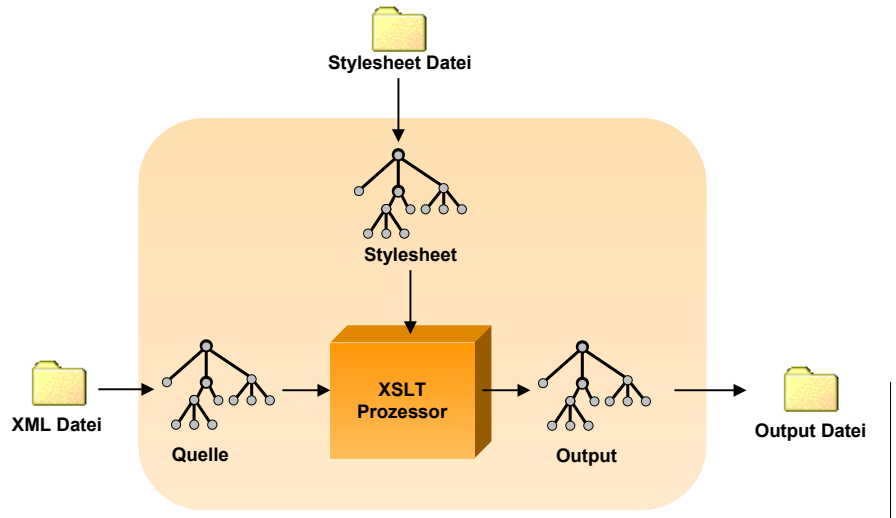


XSLT Prozessor

Orientation in Objects

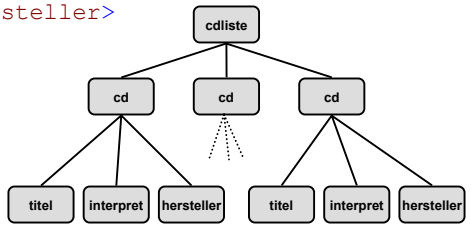


XSLT Prozessor (mit Bäumen)



XML-Quelle für die Ausführungen

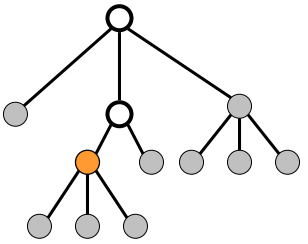
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cdliste>
   <cd id="65" jahr="2000">
        <titel>Music</titel>
        <interpret>Madonna</interpret>
        <hersteller>BMG</hersteller>
   </cd>
   <cd id="26" jahr="1995">
        <titel>Ray of light</titel>
        <interpret>Madonna</interpret>
        <hersteller>BMG</hersteller>
   </cd>
   ....
</cdliste>
```



XPath

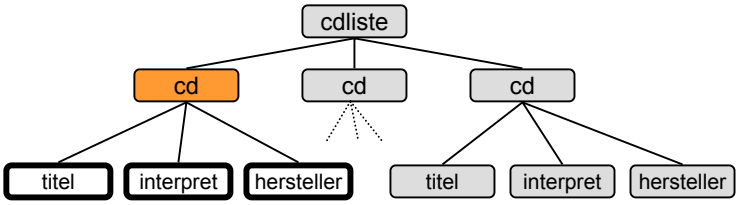
Orientation in Objects

- Sprache für das Addressieren von Teilen eines XML Dokumentes
- Wurde entworfen für XSLT und XPointer
- Enthält Funktionen für die Manipulation von Strings, Zahlen und Booleans



Beispiel: child Achse

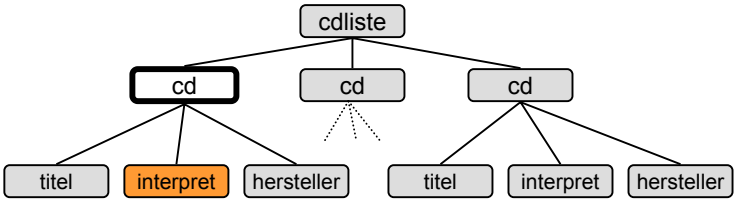
Orientation in Objects



Kindknoten des Context Node werden adressiert

Beispiel: parent Achse

Orientation in Objects



Jeder Knoten hat einen “parent-node”, mit Ausnahme des root-Element.

XML Patterns

Orientation in Objects

- Catch-All Element
- Choice Reducing Container
- Container Element
- Envelope
- Optional Container Element
- ...

Quelle: http://www.xmlpatterns.com/

XML Pattern: Head-Body

Orientation in Objects

```
<BMECAT version="1.2">
    <HEADER>
        <CATALOG>
            <CATALOG_ID>f:3M739</CATALOG_ID>
            <CATALOG_NAME>Kurs</CATALOG_NAME>
            ...
        </CATALOG>
        ...
    </HEADER>
    <T_NEW_CATALOG>
        <ARTICLE>
        ...
        </ARTICLE>
    </T_NEW_CATALOG>
</BMECAT>
```


Inhalt

Orientation in Objects

- Einleitung
  - XSL & XSLT
  - XPath
  - XML Pattern
- XSLT Patterns
  - XSLT Stylesheet Patterns
  - XSLT Patterns
- XSLT Praxis Techniken und Beispiele
  - Mengenoperationen
  - Techniken zu Sortieren, Gruppieren,...
  - Inhaltsverzeichnis erstellen
  - Cross References
  - Best Practise

Stylesheet Design Patterns nach Kay

- Fill-in-the-blanks Stylesheet
- Navigational Stylesheet
- Rule-Based Stylesheet
- Computational Stylesheet

(Siehe Michael Kay „XSLT Programmer's Reference“ Kapitel 8)

Fill-in-the-blanks Stylesheet

- Einfach
- HTML mit XSLT Befehlen
- Stylesheet gleicht in der Struktur dem Output

Fill-in-the-blanks Stylesheet: Beispiel

Orientation in Objects

```
<html xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xsl:version="1.0">
  <head>
    <title>CD Liste</title>
  </head>
  <body>
    <xsl:for-each select="cdliste/cd">
      <b><xsl:value-of select="interpret"/></b>
      :<xsl:value-of select="titel"/><br/>
    </xsl:for-each>
  </body>
</html>
```


Fill-in-the-blanks Stylesheet: Ausgabe

Orientation in Objects

Ausgabe mit cdliste.xml:

Madonna: Music
Madonna: Ray of light
Eminem: Marshall Mathers
Eminem: Eminem
Britney Spears: baby one more time

Navigational Stylesheet

Orientation in Objects

- Fortführung des Fill-in-the-blanks Stylesheet
- Output orientiert
- Kann enthalten
  - Named Templates, Variablen, keys, Conditionals, ...
- Gleicht prozeduralem Programm
- Enthält xsl:stylesheet und xsl:template Tags
- Es wird vorgegeben, wo die Knoten im Source gefunden werden können
- Erfordert starre Struktur des Inputs

Navigational Stylesheet: Beispiel

Orientation in Objects

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">

<xsl:template match="/">
  <html><body>
    <xsl:for-each select="cdliste/cd">
      <br/><b><xsl:value-of select="interpret"/></b>
      <xsl:call-template name="getTitel">
        <xsl:with-param name="id" select="@id"/>
      </xsl:call-template>
    </xsl:for-each>
  </body></html>
</xsl:template>

<xsl:template name="getTitel">
   <xsl:param name="id"/>
   <xsl:text> </xsl:text><xsl:value-of
   select="//cd[@id=$id]/titel"/>
</xsl:template>

</xsl:stylesheet>
```

Template

Orientation in Objects

- Besteht aus zwei Bestandteilen
  - Ein Pattern, welches auf Knoten des Sourcetrees angewandt wird
  - Ein Template, welches instanziiert werden kann, damit es Bestandteil des Resulttrees wird

Matchpattern

```
<xsl:template match="hersteller">
        <b>Hersteller:</b>
        <xsl:value-of select="."/>
        <br/>
</xsl:template>
```


Ablauf des Processing Model

Orientation in Objects

- Liste von Knoten wird der Reihenfolge nach verarbeitet
- Für den Knoten werden passende Template Rules gesucht
- Die spezifischste Regel wird ausgewählt
- Das Template wird instanziiert
- Der entsprechende Knoten wird zum Context Knoten

Rule-Based Stylesheet

Orientation in Objects

- Besteht vorwiegend aus Regeln, wie bestimmte Dinge aus der Quelle verarbeitet werden sollen
- Macht nur wenige Vermutungen über Quelle und Output
- Anwendung: Bei wenig bekannter oder flexibler Struktur der Quelle
- Teile können gut wiederverwendet werden

Rule-Based Stylesheet

Orientation in Objects

```
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">

<xsl:template match="/">
  <html><body>
    <xsl:apply-templates/>
  </body></html>
</xsl:template>

<xsl:template match="interpret">
   <b><xsl:value-of select="."/></b><br/>
</xsl:template>

<xsl:template match="titel">
   <xsl:value-of select="."/>
</xsl:template>

</xsl:stylesheet>
```

Computational Stylesheet

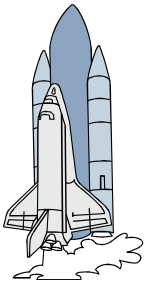
Orientation in Objects

- Struktur der Quelle stimmt nicht mit Output überein
- Anwendung:
  - Umformatierung, Aggregation
- Verwendet Anteile der funktionalen Programmierung

Exkurs: Funktionale Programmierung

Orientation in Objects

- Ohne Zuweisungen
- Zuweisungen erzwingen bestimmten Programmablauf
- Seiteneffekte werden vermieden
- Rekursionen ersetzen Schleifen



Computational Stylesheet: Beispiel

Orientation in Objects

Quelle:

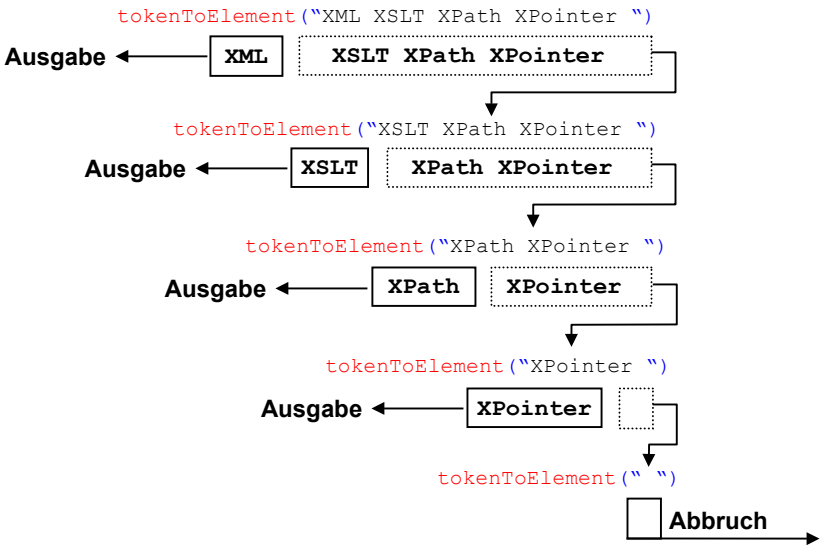
```
<keywords>XML XSLT XPath XPointer</keywords>
```

Output:

```
<keyword>XML</keyword>
<keyword>XSLT</keyword>
<keyword>XPath</keyword>
<keyword>XPointer</keyword>
```


Computational Stylesheet: Beispiel Ablauf

Orientation in Objects



Computational Stylesheet: Beispiel

Orientation in Objects

Rekursives Template

```
<xsl:template match="keywords" name="tokenToElement">
  <xsl:param name="words" select="concat(.,' ')"/>
  <xsl:choose>
    <xsl:when test="string-length($words) > 1">
      <keyword>
        <xsl:value-of select="substring-before($words,' ')"/>
      </keyword>
      <xsl:call-template name="tokenToElement">
        <xsl:with-param name="words"
         select="substring-after($words,' ')"/>
      </xsl:call-template>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```


Rekursion: Fakultät berechnen

Orientation in Objects

```
<fakultaet>5</fakultaet>
```

5! = 5 * 4 * 3 * 2 * 1

5! = 5 * (4!)
5! = 5 * 4 * (3!)
5! = 5 * 4 * 3 * (2!)
5! = 5 * 4 * 3 * 2 * (1!)
5! = 5 * 4 * 3 * 2 * 1

Rekursion: Fakultät berechnen

Orientation in Objects

```
<xsl:template name="fakultaet" match="fakultaet">
  <xsl:param name="zahl" select="."/>
  <xsl:choose>
    <xsl:when test="$zahl=0">1</xsl:when>
    <xsl:otherwise>
      <xsl:variable name="fak">
        <xsl:call-template name="fakultaet">
          <xsl:with-param name="zahl" select="$zahl - 1"/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:value-of select="$zahl * $fak"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

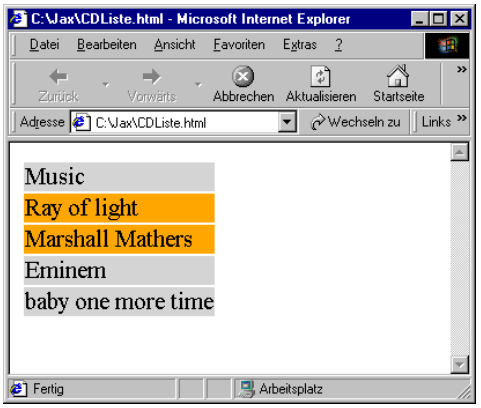

Inhalt

Orientation in Objects

- Einleitung
  - XSL & XSLT
  - XPath
  - XML Pattern
- XSLT Patterns
  - XSLT Stylesheet Patterns
  - XSLT Patterns
- XSLT Praxis Techniken und Beispiele
  - Mengenoperationen
  - Techniken zu Sortieren, Gruppieren,...
  - Inhaltsverzeichnis erstellen
  - Cross References
  - Best Practise

Aufgabe: CDs mit Preis kennzeichnen

Orientation in Objects



Lösung: Condition & Variable

Orientation in Objects

```
<xsl:template match="cd">

  <xsl:variable name="farbe">
    <xsl:choose>
      <xsl:when test="preis">orange</xsl:when>
      <xsl:otherwise>lightgrey</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <tr style="background: {$farbe}">
    <td><xsl:value-of select="titel"/></td>
  </tr>

</xsl:template>
```

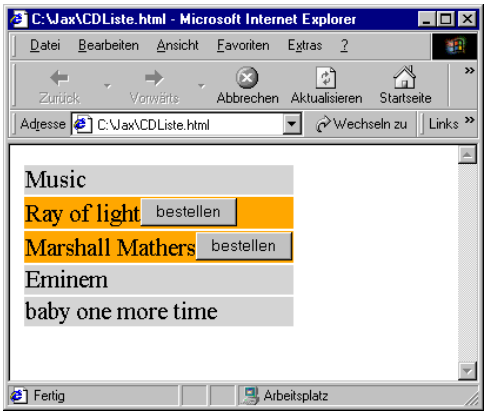
Lösung: Condition

Orientation in Objects

```
<xsl:template match="cd">
  <tr>
    <xsl:attribute name="style">
      background:
      <xsl:choose>
        <xsl:when test="preis">orange</xsl:when>
        <xsl:otherwise>lightgrey</xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
    <td>
      <xsl:value-of select="titel"/>
    </td>
  </tr>
</xsl:template>
```


Aufgabe: Zusätzlicher Knopf

Orientation in Objects



Lösung: Mehrere Conditions

Orientation in Objects

```
<xsl:template match="cd">
  <tr>
    <xsl:attribute name="style">
      background:
      <xsl:choose>
        <xsl:when test="preis">orange</xsl:when>
        <xsl:otherwise>lightgrey</xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>
    <td>
      <xsl:value-of select="titel"/>
      <xsl:if test="preis">
        <input type="button" value="bestellen"/>
      </xsl:if>
    </td>
  </tr>
</xsl:template>
```


Lösung: Overloaded Templates

Orientation in Objects

```
<xsl:template match="cd">
  <tr style="background: lightgrey">
    <td><xsl:value-of select="titel"/></td>
  </tr>
</xsl:template>

<xsl:template match="cd[preis]">
  <tr style="background: orange">
    <td>
      <xsl:value-of select="titel"/>
      <input type="button" value="bestellen"/>
    </td>
  </tr>
</xsl:template>
```

Pattern: Overloaded Templates

Orientation in Objects

- Abstrakt
  - Aufteilung von unübersichtlichen Templates mit Bedingungselementen auf mehrere einzelne Templates, die um ein zusätzliches Prädikat erweitert werden.
- Motivation
  - Bedingungselemente in Templates können unübersichtlich und schlecht wartbar sein.
- Anwendbarkeit
  - Verwendung wenn verwandte Matchpattern sich in der Instanziierung unterscheiden sollen.
  - Verwendung für die übersichtliche Behandlung von Sonderfällen
  - Verwendung wenn unterschiedliche Verhaltensweisen durch Bedingungsanweisungen behandelt werden

Pattern: Overloaded Templates

Orientation in Objects

- Struktur
  - Mehrere Templates unterscheiden sich im Matchpattern nur durch ein zusätzliches Prädikat. Ein zusätzliches Defaulttemplate kann durch Weglassen des Prädikates definiert werden.
- Teilnehmer
  - Mehrere alternative Templates, die sich im Matchpattern bis auf ein Prädikat unterscheiden
  - Aufrufendes Template, welches die alternativen Regeln durch ein apply-templates Element aufruft. Der XPath Ausdruck im select ist gegenüber dem zusätzlichen Prädikat unspezifisch.

Pattern: Overloaded Templates

Orientation in Objects

- Konsequenzen und Diskussion
  - Mehr Templates
  - Dafür kleinere und übersichtlichere Templates
  - Unter Umständen doppelter Code
  - Auf unübersichtliche Bedingungselemente (choose, if) kann verzichtet werden, der Code in den einzelnen Templates wird übersichtlicher
- Anwendungen
  - Sonderbehandlungen in bestimmten Fällen z.B. Limit überschritten, Hervorhebungen

Pattern: Overloaded Templates

Orientation in Objects

- Beispielcode: Alternierende Farben

```
<xsl:template match="cd">
   <tr style="background: lightgrey">
       <td><xsl:value-of select="titel"/></td>
   </tr>
</xsl:template>

<xsl:template match="cd[position() mod 2 = 1]">
   <tr style="background: orange">
       <td><xsl:value-of select="titel"/></td>
   </tr>
</xsl:template>
```

Pattern: Overloaded Templates

Orientation in Objects

- Bekannte Verwendung
  - Cocoon 2(a5) verwendet im Core Stylesheet für das Erstellen der Sitemap das Pattern. Der Javacode für Actions wird je nach dem, ob ein Actionset verwendet wird von einem unterschiedlichen Template erzeugt.

```
<!--processing of an act element having a type attribute-->
<xsl:template match="map:act[@type]">
  ...
</xsl:template> <!-- match="map:act[@type]" -->

<!--processing of an act element having a set attribute-->
<xsl:template match="map:act[@set]">
  ...
</xsl:template> <!-- match="map:act[@set]" -->
```

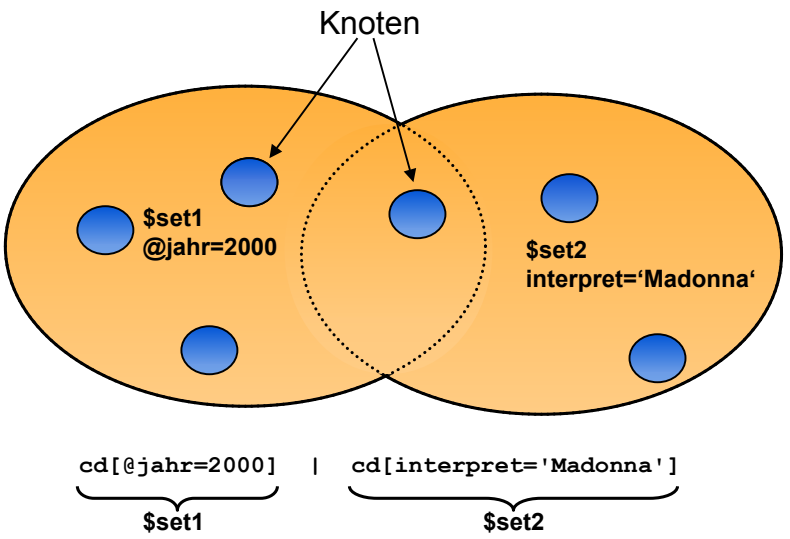

Inhalt

Orientation in Objects

- Einleitung
  - XSL & XSLT
  - XPath
  - XML Pattern
- XSLT Patterns
  - XSLT Stylesheet Patterns
  - XSLT Patterns
- XSLT Praxis Techniken und Beispiele
  - Mengenoperationen
  - Techniken zu Sortieren, Gruppieren,...
  - Inhaltsverzeichnis erstellen
  - Cross References
  - Best Practise

Mengen: Vereinigung

Orientation in Objects




```
cd[@jahr=2000]   |   cd[interpret='Madonna']
```

$set1 $set2

Mengen: Schnittmenge (Kaysian Technique)

Orientation in Objects

- **Problem:** Nur die Vereinigung ist standardisiert
- Schnittmenge und Differenz bei manchen Implementierungen als erweiterte Funktionalität

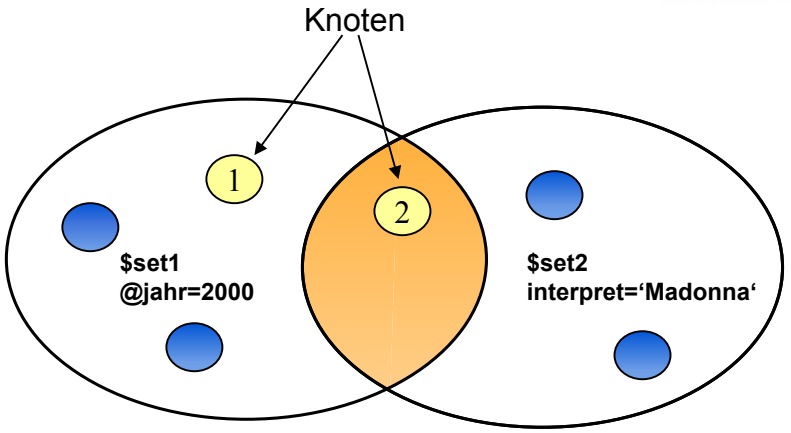
Idee:

```
count($nodeset) = count($nodeset | $node)
```

True falls $node Member im Set $nodeset ist

Mengen: Schnittmenge (Kaysian Technique)

Orientation in Objects




```
$set1[count(.|$set2) = count($set2)]
```

temp set

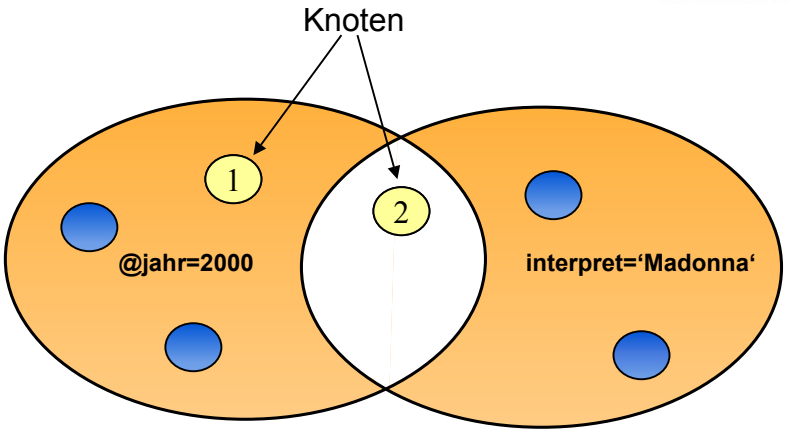
Mengen: Schnittmenge (Kaysian Technique)

Orientation in Objects

```
<xsl:variable name="set1" select="cdliste/cd[@jahr=2000]"/>

<xsl:variable name="set2"
              select="cdliste/cd[interpret='Madonna']"/>

...

<xsl:for-each select="$set1[count(.|$set2) = count($set2)]">
   <xsl:value-of select="titel"/><br/>
</xsl:for-each>

...
```

Mengen: Symmetrische Differenz (Ken Holman)

Orientation in Objects




```
$set1[count(.|$set2)!=count($set2)] |
$set2[count(.|$set1)!=count($set1)]
```


Sortieren

Orientation in Objects

Kann bei for-each und apply-templats angewandt werden

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

Beispiel: Uhrzeiten

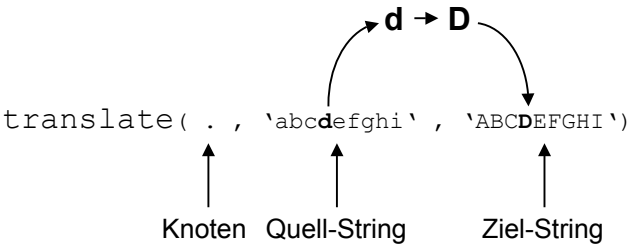
```
<zeit>17:30</zeit>
<zeit>8:23</zeit>
<zeit>15:50</zeit>
```

```
<xsl:sort select="substring-before(zeit,':')"/>
<xsl:sort select="substring-after(zeit,':')"/>
```

Sortieren: Ignore-Case Sortierung

Orientation in Objects

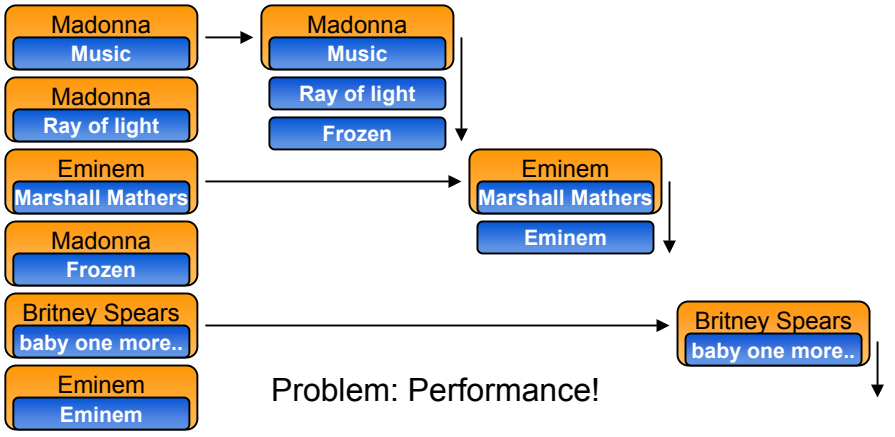
```
<xsl:sort
   select="translate(.,'abcdefghijklmnopqrstvwxyz','ABCD
   EFGHIJKLMNOPQRSTVWXYZ')"/>
```



Gruppieren in XSL

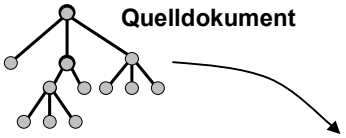
Orientation in Objects

```
<xsl:for-each select="//interpret[not(.= preceding::interpret)]">
<xsl:for-each select="//CD[interpret=current()]">
```



Keys: Key-Tabellen definieren

Orientation in Objects



Quelldokument

```
<xsl:key name="cds" match="cd" use="interpret"/>
```

Stylesheet

“cds“ Schlüsseltabelle

Schlüssel	**Knoten**
Madonna	cd
Madonna	cd
Eminem	cd
Eminem	cd
Britney Spears	cd
Eminem	cd
...	cd

Keys: Key-Tabellen verwenden

Orientation in Objects

Beispiel: Schleife über Nodeset

Nodeset mit allen
Knoten zum Schlüssel

```
<xsl:for-each select="key('cds', 'Madonna')">
...
</xsl:for-each>
```

Name der Key-Tabelle

gesuchter String

Keys: Vor- und Nachteile

Orientation in Objects

- **Vorteile**
  - Findet alle Knoten zu einem festgelegten Schlüssel
  - Zugriff über Key
  - Indizierung der Tabelle steigert Geschwindigkeit
  - Keys sind schneller als XPath-Ausdrücke und Prädikate
- **Nachteile**
  - Speicherintensiv

Identifikation: `generate-id()`

Orientation in Objects

- *`generate-id()`* liefert zu einem Knoten eine eindeutige ID

```
<xsl:value-of select="generate-id(.)"/>
```

- Ausgabe bei Saxon:

```
Music(b1b1b2)
Ray of light(b1b1b4)
Marshall Mathers(b1b1b6)
Eminem(b1b1b8)
baby one more time(b1b1c10)
```

Identifikation: Erstes Element finden

Orientation in Objects

- Bilden der eigenen id
- Vergleich mit Eintrag aus der Key-Tabelle
- Nur erster Eintrag wird angezeigt

```
cd[generate-id(.)=generate-id(key('cds', interpret))]
```

Muench´sche Methode: Beispiel

Orientation in Objects

```
<xsl:key name="cds" match="cd" use="interpret"/>

<xsl:template match="/">
   <xsl:for-each select="//cd[generate-id(.)=
                 generate-id(key('cds', interpret))]">
      <xsl:value-of select="interpret"/><br/>
      <xsl:for-each select="key('cds', interpret)">
        Titel: <xsl:value-of select="titel"/><br/>
        Hersteller: <xsl:value-of select="hersteller"/><br/>
      </xsl:for-each>
      <br/>
   </xsl:for-each>
</xsl:template>
```

Muench´sche Methode: Ausgabe


```
Madonna
Titel: Music
Hersteller: BMG
Titel: Ray of light
Hersteller: BMG

Eminem
Titel: Marshall Mathers
Hersteller: Sony
Titel: Eminem
Hersteller: Sony

Britney Spears
Titel: baby one more time
Hersteller: Sony
```


Muench´sche Methode: Vor- und Nachteile

- **Vorteile**
  - Kein mehrfaches Durchsuchen
  - Gute Performance
  - Verarbeitung von großen Quelldokumenten
  - Kann flache Struktur in eine hierarchische abbilden
- **Nachteile**
  - Kann durch Key-Tabelle speicherintensiv werden

Erzeugen eines Inhaltsverzeichnisses

Orientation in Objects

Beispiel: Eine Büchersammlung als docbook-Dokument

```
<set>
  <book>
  <title> My first book</title>
    <chapter><title> Beginning</title>
      <section><title> First Subtitle</title>
        <para>...</para>
      </section>
      <section><title> Second Subtitle</title>
        <para>...</para>
      </section>
    </chapter>
  </book>
</set>
```


Erzeugen eines Inhaltsverzeichnisses

Orientation in Objects

Das zugehörige XSL-Dokument

```
...
<xsl:template match="/">
  <h2>Directory</h2>
  <xsl:apply-templates select="//title" mode="index"/>
</xsl:template>

<xsl:template match="title" mode="index">
  <xsl:number level="multiple" format="1.1.1"
  count=" book | chapter | section " /><br/>
  <xsl:value-of select="."/>
</xsl:template>
```

Cross-references: XML-Quelle Büchersammlung

Orientation in Objects

```
...
<chapter>
  <title id="start">Beginning</title>
    How all began is a very interesting story...
    <pointer target="intro"/>
  </para>
</chapter>
<chapter>
  <title id="intro">Introduction</title>
    <para>
    Here, I will introduce you some important things...
    </para>
  </chapter>
  ...
```


Cross-references: Das XSL-Stylesheet

Orientation in Objects

```
...
<xsl:template match="pointer">
  See Section <xsl:apply-templates mode="crossref"
  select="id(@target)"/> for further information.<br/>
</xsl:template>

...

<xsl:template match="chapter | title | para" mode="crossref">
  <b><xsl:value-of select="."/></b>
</xsl:template>
```

Best Practise: Schleifen

Orientation in Objects

Mit Condition:

```
<xsl:for-each select="cd">
   <xsl:if test="@jahr=2000">
        <xsl:value-of select="titel"/><br/>
   </xsl:if>
</xsl:for-each>
```

Besser mit Predikat:

```
<xsl:for-each select="cd[@jahr=2000]">
  <xsl:value-of select="titel"/><br/>
</xsl:for-each>
```


Best Practise: Dynamische Attribute

Orientation in Objects

Mit Variable:

```
<xsl:variable name="path">
  <xsl:call-template name="getDirPath">
    <xsl:with-param name="node"
         select="key('seiten',@name)"/>
  </xsl:call-template>
</xsl:variable>

<a href="{$path}">Link</a>
```

Best Practise: Dynamische Attribute

Orientation in Objects

Besser mit Attribut-Tag:

```
<a>
  <xsl:attribute name="href">
    <xsl:call-template name="getDirPath">
     <xsl:with-param name="node" select="key('seiten',@name)"/>
    </xsl:call-template>
    Link
  </xsl:attribute>
</a>
```

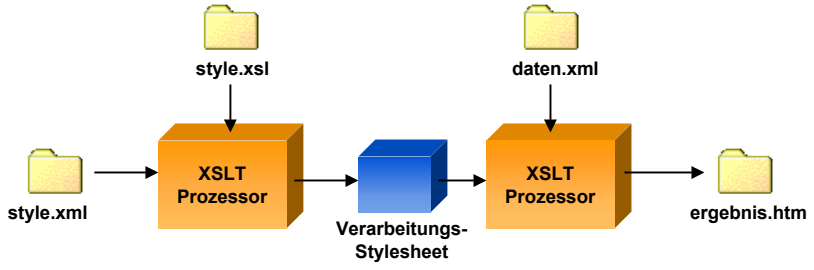

Problem: Dynamische Templates

Orientation in Objects

Problem

```
<xsl:call-template name="$blub"/>
```

Lösung: Stylesheet erzeugt Stylesheet



Quellen

Orientation in Objects

- Specification XSL 1.0 (candidate recommendation)
  http://www.w3.org/TR/xsl/
- Specification XSLT 1.0 (recommendation)
  http://www.w3.org/TR/xslt.html
- Specification XSLT 1.1 (working draft)
  http://www.w3.org/TR/xslt11/
- Specification XPath 1.0 (recommendation)
  http://www.w3.org/TR/xpath.html
- Specification XML 1.0
  http://www.w3.org/XML/
- **XSLT Questions and Answers**
  http://www.dpawson.co.uk/xsl/sect21.html

Quellen

Orientation in Objects

- XSLT Benchmark
  http://www.xml.com/pub/a/2001/03/28/xsltmark/results.html
- Crane Softwrights Ltd.
  http://www.cranesoftwrights.com/resources/xslkeys/index.htm
- Michael Key, XSLT - Programmer´s Reference
  WROX
- Oliver Becker, XSLT-Tutorial
  www.informatik.hu-berlin.de/~obecker/XSLT/Tutorials/muenchian
- XSLT-Tutorial ZVON.org
  http://www.zvon.org
- XML Patterns.com
  http://www.xmlpatterns.com/

Fragen

Orientation in Objects

Orientation in Objects

Vielen Dank für die Aufmerksamkeit!

Thomas Bayer
bayer@oio.de