Gliederung

- XSLT
- XQuery

XSLT

- XSL(T) steht für EXtensible Stylesheet Language (Transformation)
- XSLT transformiert ein XML Dokument in eine anderes XML Dokument
- XSLT nutzt XPath zum navigieren in XML Dokumenten
- Browser Support für XML und XSLT
  - Unterschiedliche Unterstützung
  - Siehe http://www.w3schools.com/browsers

XSLT Beispiel

- **XML Datei**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
 <artist>Bob Dylan</artist>
 <country>USA</country>
<company>Columbia</company>
 <price>10.90</price>
 <year>1985</year>
 </cd>
. . .
</catalog>
```

- **Verbindung von XML und XSLT**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"
    href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
...
```

- **XSLT Datei**

```
<?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>
<h2>My CD Collection</h2>
<table border="1">
  <tr bgcolor="#9acd32">
    <th align="left">Title</th>
    <th align="left">Artist</th>
  </tr>
<xsl:for-each select="catalog/cd">
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
</table> </body> </html>
</xsl:template> </xsl:stylesheet>
```


XSLT Elemente: Templates

- <xsl:template> Element
  - Definiert ein Template
  - match Attribut verbindet das Template mit einem XML Element
- Vereinfachtes Beispiel

```
<?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>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th> <th>Artist</th>
    </tr>
    <tr> <td>.</td> <td>.</td> </tr>
  </table>
</body> </html>
</xsl:template> </xsl:stylesheet>
```

XSLT Element: <xsl:value-of>

- <xsl:value-of> extrahiert den Wert eines XML Elements

```
<?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>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th> <th>Artist</th>
    </tr>
     <tr>
        <td><xsl:value-of select="catalog/cd/title"/> </td>
        <td><xsl:value-of select="catalog/cd/artist"/></td>
</tr>
  </table>
</body> </html>
</xsl:template> </xsl:stylesheet>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan

XSLT Element: <xsl:for-each>

- <xsl:for-each> selektiert alle XML Elemente einer mit XPath beschriebenen Knotenmenge

```
<?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>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32"> <th>Title</th>
      <th>Artist</th> </tr>
<xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="catalog/cd/title"/> </td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
    </tr>
</xsl:for-each>
  </table>
</body> </html>
</xsl:template> </xsl:stylesheet>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary More
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge

XSLT Element: <xsl:sort>

- <xsl:sort> innerhalb von <xsl:for-each> sortiert die selektierten Knoten

```
<?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>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32"><th>Title</th><th>Artist</th></tr>
<xsl:for-each select="catalog/cd">
<xsl:sort select="artist"/>
    <tr>
      <td><xsl:value-of select="catalog/cd/title"/> </td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
    </tr>
</xsl:for-each>
  </table>
</body> </html>
</xsl:template> </xsl:stylesheet>
```

My CD Collection

Title	Artist
Romanza	Andrea Bocelli
One night only	Bee Gees
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
The very best of	Cat Stevens
Greatest Hits	Dolly Parton
Sylvias Mother	Dr.Hook
Eros	Eros Ramazzotti
Still got the blues	Gary Moore
Unchain my heart	Joe Cocker
Soulsville	Jorn Hoel

XSLT Element: <xsl:if>

- <xsl:if> testet den Inhalt der XML Daten
- Syntax:
  - <xsl:if test="*expression*"> ...Ausgabe, falls Test wahr ist ...</xsl:if>

```
<?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> <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32"><th>Title</th><th>Artist</th></tr>
<xsl:for-each select="catalog/cd">
<xsl:if test="price &gt; 10">
    <tr>
      <td><xsl:value-of select="catalog/cd/title"/> </td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
    </tr>
</xsl:if>
</xsl:for-each>
</table></body> </html>
</xsl:template> </xsl:stylesheet>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Still got the blues	Gary Moore
One night only	Bee Gees
Romanza	Andrea Bocelli
Black Angel	Savage Rose
1999 Grammy Nominees	Many

XSLT Element: <xsl:choose>

- <xsl:choose> in Verbindung mit <xsl:when> und <xsl:otherwise> drückt mehrere alternative Bedingungen aus.
- Syntax
  - <xsl:choose>
    <xsl:when test="*expression*">
    ... some output ...
    </xsl:when>
    <xsl:otherwise>
    ... some output ....
    </xsl:otherwise>
    </xsl:choose>

XSLT Element: <xsl:choose>

```
<?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> <h2>My CD Collection</h2>
 <table border="1">
  <tr bgcolor="#9acd32"><th>Title</th><th>Artist</th></tr>
<xsl:for-each select="catalog/cd">
  <tr>
    <td><xsl:value-of select="catalog/cd/title"/> </td>
    <xsl:choose>
      <xsl:when test="price &gt; 10">
        <td bgcolor="#ff00ff"> <xsl:value-of select="artist"/></td></xsl:when>
      <xsl:when test="price &gt; 9">
        <td bgcolor="#cccccc"> <xsl:value-of select="artist"/></td> </xsl:when>
      <xsl:otherwise><td><xsl:value-of select="artist"/></td> </xsl:otherwise>
    </xsl:choose>
  </tr>
</xsl:for-each>
</table></body> </html>
</xsl:template> </xsl:stylesheet>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith

Beachte: <xsl:choose> wählt den ersten erfolgreichen Test ab

XSLT Element: <xsl:apply-templates>

- <xsl:apply-templates> wendet ein Template auf das akt. Element oder Kinder des akt. Elements an.
- Falls das select Attribut dabei ist, werden nur die passenden Elemente verarbeitet.
- Dadurch kann die Verarbeitungsreihenfolge gesteuert werden.


```
<?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>
<h2>My CD Collection</h2>
<xsl:apply-templates/>
</body> </html>
</xsl:template>

<xsl:template match="cd">
<p>
<xsl:apply-templates select="title"/>
<xsl:apply-templates select="artist"/>
</p>
</xsl:template>

<xsl:template match="title">
Title: <span style="color:#ff0000"><xsl:value-of select="."/></span> <br />
</xsl:template>

<xsl:template match="artist">
Artist: <span style="color:#00ff00"> <xsl:value-of select="."/></span> <br />
</xsl:template>
</xsl:stylesheet>
```

My CD Collection

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

XSLT Zusammenfassung

- XSLT Elemente
  - siehe: http://www.w3schools.com/xsl/xsl_w3celementref.asp
- XSLT Funktionen
  - XPath (http://www.w3.org/2005/02/xpath-functions)
  - Plus weitere Funktionen (http://www.w3schools.com/xsl/xsl_functions.asp)
    - current() Returns the current node
    - document() Used to access the nodes in an external XML document
    - element-available() Tests whether the element specified is supported by the XSLT processor
    - format-number() Converts a number into a string
    - function-available() Tests whether the function specified is supported by the XSLT processor
    - generate-id() Returns a string value that uniquely identifies a specified node
    - key() Returns a node-set using the index specified by an <xsl:key> element
    - system-property() Returns the value of the system properties
    - unparsed-entity-uri() Returns the URI of an unparsed entity

XQuery

- XQuery ist die Anfragesprache für XML Daten
- XQuery ist für XML was SQL für Datenbanken ist
- XQuery baut auf XPath Ausdrücken auf
- XQuery wird von alle großen DBs unterstützt (IBM, Oracle, Microsoft, etc.)
- XQuery 1.0 und XPath 2.0 teilen das gleiche Datenmodell und unterstützen die gleichen Funktionen und Operatoren.
- XQuery kann genutzt werden zum
  - Extrahieren von Informationen um in Web Services weiterverwendet werden
  - Generieren von Zusammenfassungen
  - Transformieren von XML Daten nach XHTML
  - Durchsuchen von Web Dokumenten nach relevanten Informationen

Beispiel: books.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>

<book category="COOKING">
<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>

<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>

<book category="WEB">
<title lang="en">XQuery Kick Start</title>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan
    Nagarajan</author>
 <year>2003</year>
<price>49.99</price>
</book>

<book category="WEB">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>

</bookstore>
```


Anfragen

- öffnen von books.xml
  - doc("books.xml")
- alle Titel selektieren
  - doc("books.xml")/bookstore/book/title
  - <title lang="en">Everyday Italian</title>
    <title lang="en">Harry Potter</title>
    <title lang="en">XQuery Kick Start</title>
    <title lang="en">Learning XML</title>
- Prädikate ausnutzen
  - doc("books.xml")/bookstore/book[price<30]
  - <book category="CHILDREN">
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
    </book>

FLWOR: For, Let, Where, Order by, Return

- XPath
  - doc("books.xml")/bookstore/book[price>30]/title
- FLWOR
  - for $x in doc("books.xml")/bookstore/book
    where $x/price>30
    return $x/title
- Mit Sortierung
  - for $x in doc("books.xml")/bookstore/book
    where $x/price>30
    order by $x/title
    return $x/title

XML nach XHTML

- FLWOR Anfrage
  - for $x in doc("books.xml")/bookstore/book
    order by $x/title
    return $x/title
- Mit XHTML Code
  - {
    for $x in doc("books.xml")/bookstore/book/title
    order by $x
    return {$x}
    }
  -
    <title lang="en">Everyday Italian</title>
    <title lang="en">Harry Potter</title>
    <title lang="en">Learning XML</title>
    <title lang="en">XQuery Kick Start</title>

- Ohne XML Title Tag
  - {
    for $x in doc("books.xml")/bookstore/book/title
    order by $x
    return {data($x)}
    }
  -
    Everyday Italian
    Harry Potter
    Learning XML
    XQuery Kick Start

Terminologie

- Knotenarten:
  - Elemente, Attribute, Text, Namespace, Processing-instruktion, Kommentar und Dokument (root) Knoten
  - <bookstore> (Dokumentknoten)
    <author>J K. Rowling</author> (Elementknoten)
    lang="en" (Attributknoten)
- Beziehungen zwischen Knoten
  - Eltern
  - Kinden
  - Geschwister
  - Vorfahren
  - Nachkommen

Xquery Syntaxregeln

- XQuery ist Case Sensitiv
- XQuery Elemente, Attribute und Variablen müssen gültige XML Namen sein
- Ein XQuery Stringwert kann in einfachen oder doppelten Anführungszeichen sein
- Eine XQuery Variable ist durch $ gefolgt von einem Name gekennzeichnet, z.B. $bookstore
- XQuery Kommentare sind durch (: und :) begrenzt, z.B. (: XQuery Comment :)
- XQuery Conditional Expressions
  - for $x in doc("books.xml")/bookstore/book
    return if ($x/@category="CHILDREN")
    then <child>{data($x/title)}</child>
    else <adult>{data($x/title)}</adult>
- XQuery Vergleiche
  - Allgemein: =, !=, <, <=, >, >=
  - Wert Vergleiche: eq, ne, lt, le, gt, ge
  - Beispiele
    - $bookstore//book/@q > 10
      Wahr, falls es ein q Attribute gibt mit Wert größer als 10.
    - $bookstore//book/@q gt 10
      Wahr, falls es genau ein q Attribut durch den Ausdruck zurückgeliefert wird und dessen Wert ist größer als 10. Falls mehrere q Attribute zurückgeliefert werden, tritt ein Fehler auf

Ergebnisse um Elemente und Attribute erweitern

- XQuery Anfrage
  - for $x in doc("books.xml")/bookstore/book
    order by $x/title
    return $x/title
- XQuery mit HTML
  - {
    for $x in doc("books.xml")/bookstore/book
    order by $x/title
    return <li class="{data($x/@category)}" >{data($x/title)}.
    Category: {data($x/@category)}
    }
  -
    <li class="COOKING">Everyday Italian. Category: COOKING
    <li class="CHILDREN">Harry Potter. Category: CHILDREN</
    <li class="WEB">Learning XML. Category: WEB
    <li class="WEB">XQuery Kick Start. Category: WEB

FLWOR: For, Let, Where, Order by, Return

- for - (optional)
  - verknüpft eine Variable mit jedem Eintrag, der zurückgeliefert wird
- let - (optional)
- where - (optional)
  - spezifiziert ein Selektionskriterium
- order by - (optional)
  - spezifiziert eine Sortierung des Ergebnisses
- return
  - spezifiziert, was zurückgeliefert werden soll

For

- Verknüpft eine Variable mit jedem Eintrag, der zurückgeliefert wird
- Es kann mehrere For-Klauseln in einem FLWOR-Ausdruck geben
- Beispiel für unbedingte Schleife
  - for $x in (1 to 3) return <test>{$x}</test>
  - <test>1</test> <test>2</test> <test>3</test>
- at Schlüsselwort zum Zählen verwenden
  - for $x at $i in doc("books.xml")/bookstore/book/title return <book>{$i}. {data($x)}</book
  - <book>1. Everyday Italian</book>
    <book>2. Harry Potter</book>
    <book>3. XQuery Kick Start</book>
    <book>4. Learning XML</book>
- Kartesisches Produkt (Geschachtelte Schleifen)
  - for $x in (10,20), $y in (100,200) return <test>x={$x} and y={$y}</test>
  - <test>x=10 and y=100</test>
    <test>x=10 and y=200</test>
    <test>x=20 and y=100</test>
    <test>x=20 and y=200</test>

Let und Where

- Let
  - erlaubt Variablenzuweisungen und vermeidet so den gleichen Ausdruck mehrfach zu wiederholen
  - Die let Klausel wird nicht iteriert.
  - Beispiel:
    - let $x := (1 to 5) return <test>{$x}</test>
    - <test>1 2 3 4 5</test>
- Where
  - spezifiziert ein oder mehrere Kriterien für das Ergebnis
  - where $x/price>30 and $x/price<100

Order By und Return

- Order by
  - spezifiziert eine Sortierung des Ergebnisses
  - for $x in doc("books.xml")/bookstore/book order by $x/@category, $x/title return $x/title
  - <title lang="en">Harry Potter</title> <title lang="en">Everyday Italian</title> <title lang="en">Learning XML</title> <title lang="en">XQuery Kick Start</title>
- Return
  - spezifiziert, was zurückgeliefert werden soll

XQuery Funktionen

- XQuery 1.0, XPath 2.0 und XSLT 2.0 haben dieselbe Funktionenbibliothek
- Siehe http://www.w3schools.com/xpath/xpath_functions.asp
- Beispiele
  - <name>{uppercase($booktitle)}</name>
  - doc("books.xml")/bookstore/book[substring(title,1,5)]
  - let $name := (substring($booktitle,1,4))

Nutzerdefinierte Funktionen

- Syntax
  - declare function *prefix:function_name*($*parameter* AS *datatype*) AS *returnDatatype*
    {
    (: ...function code here... :)
    };
  - Funktionsnamen müssen ein Präfix haben
  - Datentypen sind meist in XML Schema definiert
  - Beispiel
    - declare function local:minPrice($price as xs:decimal?, $discount as xs:decimal?) AS xs:decimal?
      {
      let $disc := ($price * $discount) div 100
      return ($price - $disc)
      };
      (: Below is an example of how to call the function above :)
      <minPrice>{local:minPrice($book/price, book/discount)}</minPrice>

XQuery Zusammenfassung

- XQuery
  - Operatoren wie in XPath http://www.w3schools.com/xpath/xpath_operators.asp
  - Funktionen wie in XPath http://www.w3schools.com/xpath/xpath_functions.asp
- XQuery Datentypen wie in XSchema
  - XSD String (http://www.w3schools.com/schema/schema_dtypes_string.asp)
  - XSD Date (http://www.w3schools.com/schema/schema_dtypes_date.asp)
  - XSD Numeric (http://www.w3schools.com/schema/schema_dtypes_numeric.asp)
  - XSD Misc (http://www.w3schools.com/schema/schema_dtypes_misc.asp)