Winter School 2024

Bergische Universität Wuppertal

18.03 – 22.03.2024

Einführung in XSLT & Übung

Bastian Politycki

Foto: Nadine Sutor

BERGISCHE UNIVERSITÄT WUPPERTAL
DHa BUW
Dokument Text Edition Graduiertenkolleg 2196
IZ ED Interdisziplinäres Zentrum für Editions- und Dokumentwissenschaft
i|d|e

Übersicht / Inhalte

- XPath-Refresher
- XSLT: Was ist das? Wieso brauchen wir das?
- Ein paar Hintergrundinformationen
- Erstellung von Transformationsszenarien
- Ein paar Details: Templates, Kontrollstrukturen und mehr
- Übungen

XPath-Refresher

In aller Kürze:

- `/` – Auswahl von Elementen im XML-Baum **genau** eine Ebene tiefer
- `//` – Auswahl von Elementen im XML-Baum **beliebiger** Tiefe
- `[ ]` – Prädikat zur Einschränkung der Auswahl
- `[ ][ ]` – zwei oder mehr Prädikate sind über ein **und** verbunden
- Wichtige Funktionen:
  - `not()`, `distinct-values()`, `count()`, `substring()`, `tokenize()`, `sum()`

XPath-Refresher

Drei kurze Fragen:

- Wie viele Einträge gibt es maximal pro Sitzung?
  - `max(//div[@type = 'session']/count(div[@type = 'entry']))`
- Welche Vornamen (Personenregister) starten mit 'Jo'?
  - `distinct-values(//listPerson//forename[starts-with(., 'Jo')])`
- Wie oft wird der Ort 'O969' im Text referenziert?
  - `count(//rs[@type = 'place'][@key = 'O969'])`

Was ist XSLT?

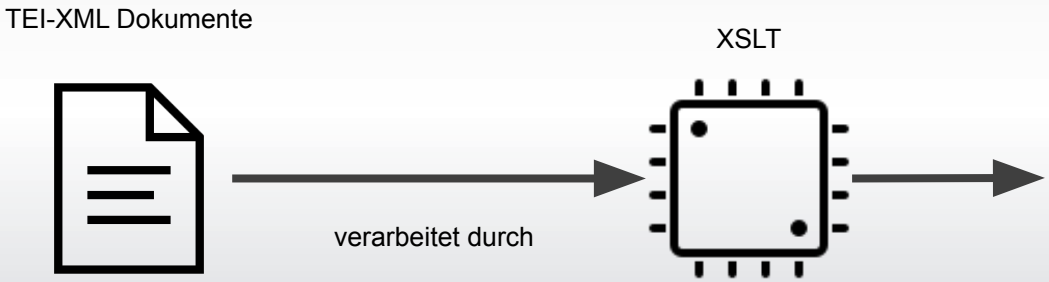
- XSLT steht für *Extensible Stylesheet Language Transformation*
- Teil von XSL (*Extensible Stylesheet Language)*
  - XML-basierte Sprachfamilie
  - weiteres Familienmitglied: XSL-FO
- Mit XSLT können Daten aus XML-Dokumenten extrahiert, umgeformt und neu strukturiert werden
- XSLT ist ein W3C-Standard:
  - 1999 => XSLT 1.0
  - 2007 => XSLT 2.0
  - 2017 => XSLT 3.0

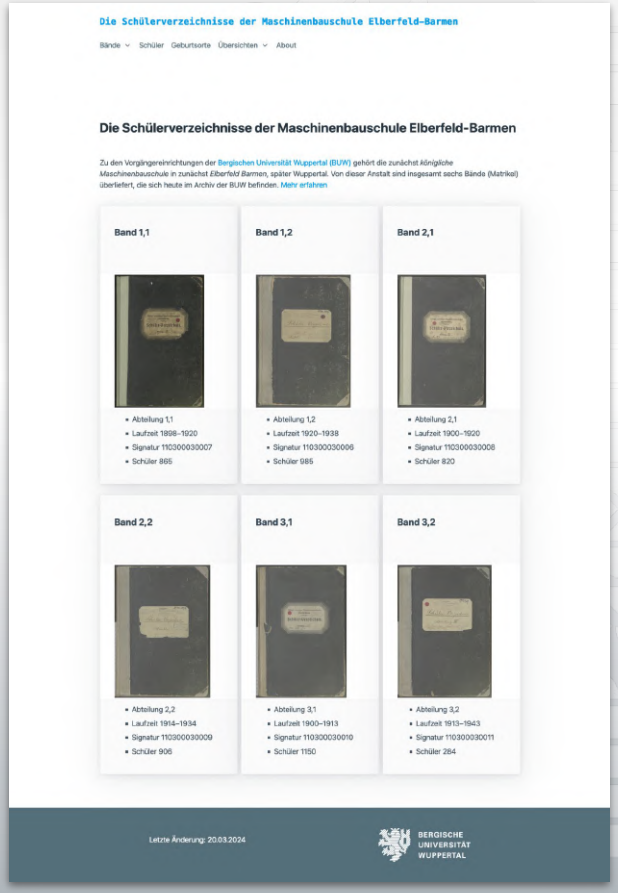
Wofür wird XSLT verwendet?

XSLT erzeugt aus einer (XML-)Struktur A eine neue Struktur B:

- Erstellung dynamischer Ausgabeformate (HTML, CSS, JS) und strukturierter Inhalte
- Aggregation, Auswertung und anschließende Ausgabe bspw. als Visualisierung
- Konvertierung in (datenbankähnliche) Formate CSV, JSON, …
- Erstellung von Berichten / Dokumentation aus 1 … n XML-Quelldaten
- Validierung / Überprüfung von XML-Daten

Wofür wird XSLT verwendet?



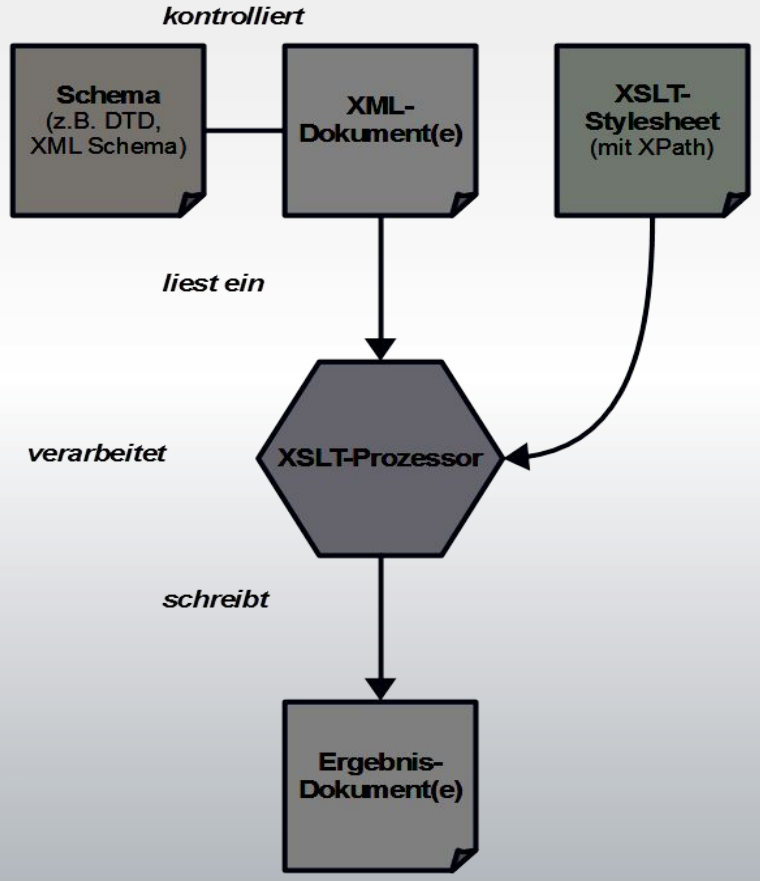


Hintergrundinformationen

XSLT…

- ist eine deklarative und funktionale Sprache
- unterstützt aber auch imperative (tue a, dann b, dann c) Programmierkonzepte
- ist (optional) statisch typisiert
- ist immutable (unveränderlich) ⇔ Skriptsprachen wie Python, JavaScript, Perl, …

Hintergrundinformationen



- Herzstück: XPath (~ 60%)
- Stylesheets werden von einem XSLT-Prozessor verarbeitet
- Eingabeformat + XSLT => Ausgabeformat

“Der Prozessor wendet die im XSLT definierten Regeln auf zur Verarbeitung der Eingabedaten an.”

Hintergrundinformationen

Funktionsweise:

- Prinzip der drei Bäume:
  - Eingabebaum (XML)
  - Verarbeitungsbaum (XSLT)
  - Ausgabebaum (bspw. HTML)
- Prinzipien der Verarbeitung:
  - Gehe von oben nach unten durch den Eingabebaum
  - Berücksichtige für jeden Knoten Anweisungen des Verarbeitungsbaums – formuliert in Templates (‚Schablonen‘) und XPath-Mustern
  - erzeuge hieraus den Ausgabebaum

Hintergrundinformationen

XSL-Template als Pseudocode:

- Definiere ein Muster, welche Teile des XML-Dokuments transformiert werden sollen
- Extrahiere Information aus diesem Teilbaum und verarbeite diese
- Erzeuge eine neue Struktur
- Füge diese Struktur der Ausgabe hinzu

Ein erstes XSLT-Skript

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
4                   exclude-result-prefixes="xs"
5                   version="3.0">
6
7       <xsl:template match="/">
8           <!-- hier geht es weiter ... -->
9       </xsl:template>
10
11  </xsl:stylesheet>
```

- 2 & 11: Wurzeltag eines XSLT-Stylesheets
- 7–9: Template, welches auf die Wurzel des XML-Eingabedokumentes ‚matcht'
- 8: Platzhalter Verarbeitungslogik

Ein erstes XSLT-Skript

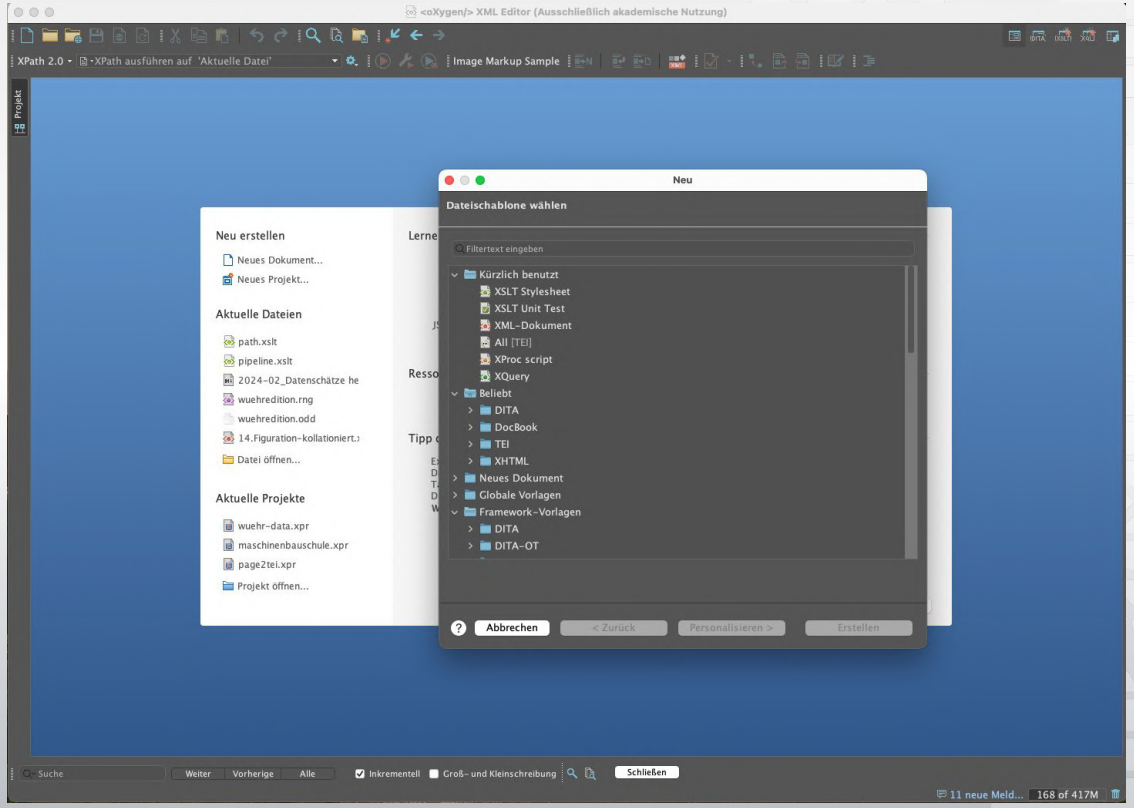
Eine kurze ‚Zwischenübung':

- XSLT-Fiddle ist eine Webanwendung, die unkompliziert erlaubt XSLT zu Testzwecken im Browser auszuführen
- Erzeugen Sie eine einfache „Hallo Welt"-Ausgabe; verwenden Sie das vorbereitete ‚Fiddle':
  - https://tinyurl.com/wsdsexsl
  - Tipp: Das XSLT-Skript enthält schon fast alles, was Sie brauchen

XSLT in Oxygen erzeugen

Neue Datei erzeugen:

- Windows: Strg+n / Mac: Cmd+N
- XSLT als Vorlage auswählen => erzeugt ein leeres XSLT Stylesheet (Sprachversion 2.0)



XSLT in Oxygen ausführen

Zwei Optionen:

- Processing Instruction im XML-Dokument: „Dieses XML-Dokument soll mit jenem XSLT-Stylesheet verarbeitet werden“
- Transformationsszenario: „Nutze jenes XSLT-Stylesheet um folgendes XML-Dokument zu verarbeiten und erzeuge dabei eine bestimmte Ausgabe(datei)“

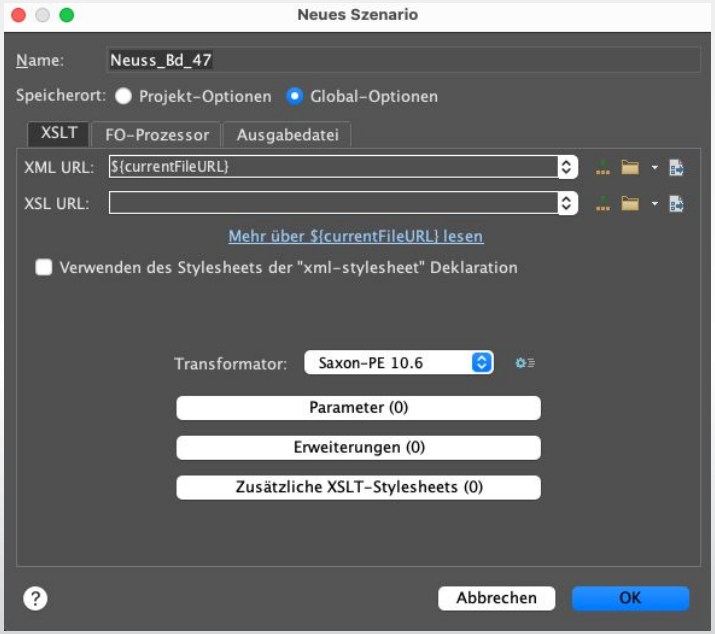
XSLT in Oxygen ausführen

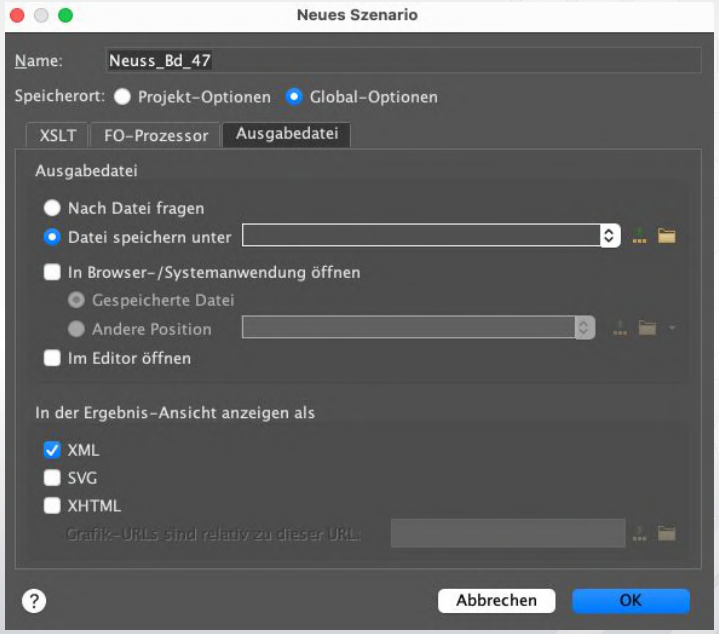
Definiertes Szenario ausführen

Szenario festlegen

Fortgeschritten: Debugger

XSLT in Oxygen ausführen





- Links: Auswahl des XSLT-Stylesheets sowie ggf. weiterer Einstellungen
- Rechts: Festlegen des Ausgabemodus (Speichern in Datei, Öffnen in Oxygen, …)

XSLT im Detail

Das Wichtigste zuerst:

- ‚Befehle' in XSLT sind auch (nur) Elemente; davon gibt es (in Saxon) für XSLT 3.0 derzeit 82
- relevant für den alltäglichen Gebrauch: ~20
- Ressourcen zum Nachschlagen:
  - data2type: *Übersicht zu XSLT*. https://www.data2type.de/xml-xslt-xslfo/xslt
  - David Birnbaum: *Digital humanities* (Abschnitt zu XSLT). http://dh.obdurodon.org/
  - Christopher Pollin: *XSLT für Digital Humanists*. https://github.com/chpollin/Teaching/tree/master/TTT/TTT_6_XSLT (+ YouTube-Playlist)
  - W3C: *XSL Transformations*. https://www.w3.org/TR/xslt/ (Spezifikationen)
  - Wilfried Grupe: *XML. Grundlagen, Technologien, Validierung, Auswertung*. Frechen 2018

XSLT im Detail

`xsl:stylesheet`	Wurzelelement eines XSL-Dokuments
`xsl:template`	Schablonen, die festlegen welche Teile des Eingangsdokuments anhand bestimmter Muster (`@match`) verarbeitet werden und wie diese in das Ergebnis einfließen
`xsl:apply-templates`	Aufruf eines Templates, dessen `match`-Attribut auf ein Kindknoten des aktuellen Kontextes passt
`xsl:value-of`	Ermöglicht die Auswahl von Werten über das Attribut `select`; diese werden als Output in das Ergebnis übertragen

Beispiel: xsl:template, xsl:apply-templates, xsl:value-of

```xml
<?xml version="1.0" encoding="utf-8"?>
<html>
  <body>
    <p>Hallo Welt</p>
  </body>
</html>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  version="3.0">

  <xsl:template match="/">
    <!-- Abgekürzte Variante von <xsl:apply-templates select="html/body/p"/> -->
    <xsl:apply-templates select="//p"/>
  </xsl:template>

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

</xsl:stylesheet>
```

Beispiel: xsl:template, xsl:apply-templates, xsl:value-of

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  version="3.0">
3
4     <xsl:template match="/">
5       <!-- Abgekürzte Variante von <xsl:apply-templates select="html/body/p"/> -->
6       <xsl:apply-templates select="//p"/>
7     </xsl:template>
8
9     <xsl:template match="p">
10      <xsl:value-of select="."/>
11    </xsl:template>
12
13  </xsl:stylesheet>
```

- 4–9: Template, welches auf die Wurzel des XML-Eingabedokumentes ‚matcht‘
- 6: Anwendung von Templates, die auf p ‚matchen‘
- 9–11: Template, welches auf p-Elemente ‚matcht‘
- 10: der eigentliche Ausgabefehl; . ist das aktuell ausgewählte Element!

XSLT im Detail

`xsl:for-each`	Führt eine Anweisung für eine Menge, die über select ausgewählt wird, durch
`xsl:for-each-group`	Ähnlich wie xsl:for-each; erlaubt jedoch über group-by zunächst Sequenzen in Gruppen zu bündeln – wichtige Funktionen: `current-group()`, `current-grouping-key()`
`xsl:sort`	Sortiert eine Sequenz
`xsl:if`	Einfache if-Abfrage, aber ohne Else!
`xsl:choose`	Ähnlich wie eine if-Abfrage nur mit `xsl:when`- und `xsl:otherwise`-Instruktionen
`xsl:variable`	definiert eine Variable, in der sich unter einem best. Namen Werte zwischenspeichern lassen
`xsl:message`	Erlaubt die “Print”-Ausgabe (z.B. für Debugging; Achtung: XSLT 3.0 notwendig

‚Real-World'-Beispiel Neusser-Projekt

- Idee:
  - Umwandlung XML → HTML
  - Ausgabe des Datums aller Sitzungen als Liste
  - ein bisschen ‚Style'
- Takeaways:
  - verschiedene Denkweisen
  - Namespace können problematisch sein
  - Text wird quasi immer verarbeitet
  - XSLT ist ein mächtiges Werkzeug gemacht für XML!
- Musterlösung im Datenordner: https://tinyurl.com/wsde2024

XSLT im Detail

Noch einmal Grundlegendes – ‚XSLT-Strategien‘:

- Pull-Ansatz:
  - Hole dir Daten aus dem Eingabedokument und tue damit etwas (`for-each`, `for-each-group`, …)
- Push-Ansatz
  - Deklarative Anweisungen (Templates): Verarbeite Daten, wenn sie ‚matchen‘ (`template`, `apply-templates`, …)

Ausblick: HTML, CSS & JS

Der Weg zur Web-Präsentation:

- HTML: Strukturierungssprache für Websiten
  - grundlegende Elemente: `HTML`, `head`, `body`, `p`, `head`, `a`
  - HTML-Crashkurs: http://dh.obdurodon.org/html_basics.xhtml
- CSS: Stylesheet-Sprache zur ‚Formatierung' von HTML (im Browser)
- JS: JavaScript => steuert Interaktionen
  - Kurzeinführung: http://dh.obdurodon.org/javascript.xhtml

XSLT-Übungen

1. Statistische Auswertungen und Ausgabe der Ergebnisse
2. Umwandlung XML => HTML
3. Anreicherung mit Registerinformationen (Daten verknüpfen, sortieren, …)
4. Wünsche…?!

XSLT-Übungen

Vorbereitung:

1. Laden Sie sich den Foliensatz herunter
2. `Neuss_Bd_47.xml` in oXygen öffnen
3. Ein neue XSLT-Datei pro Übungsaufgabe anlegen
4. Ein Transformationsszenario pro Übungsaufgabe anlegen

XSLT-Übungen

Vorbereitung:

1. Laden Sie sich den Foliensatz herunter
2. `Neuss_Bd_47.xml` in oXygen öffnen
3. Ein neue XSLT-Datei pro Übungsaufgabe anlegen (oder nutzen Sie die vorbereiteten Skripte)
4. Ein Transformationsszenario pro Übungsaufgabe anlegen

XSLT-Übungen

Übung 1:

- Nehmen Sie ein paar einfache statistische Auswertungen vor und bringen Sie deren Ergebnisse (bspw. als HTML-Seite) zur Anzeige:
  - Wie viele Sitzungen des Stadtrates gab es?
  - In welcher Sitzung wurden die meisten Einträge behandelt und wann fand diese statt?
  - Bonus: Geben Sie für jede Sitzung ‚Kopfzeile' der Sitzung aus. Sortieren Sie die Ausgabe nach Menge der Einträge.

XSLT-Übungen

Übung 2:

- Wir erarbeiten eine ‚Miniedition‘ – erstellen Sie eine HTML-Ausgabe mit folgenden Bestandteilen:
  - Ein allgemeiner Seitentitel
  - Ein Titel pro Sitzung
  - Text jeder Sitzung (Was interessiert uns? Was hat keine Relevanz? Wollen wir nur die ersten X-Zeilen berücksichtigen?)
  - Markierung von Named Entities (bspw. mit `<b>`, `<i>`, …)
  - Bonus: Inhaltsverzeichnis

XSLT-Übungen

Übung 3:

- Die Miniedition soll um ein Orts- und Personenregister ergänzt werden. Dieses besteht aus:
  - Einer sortierten Liste aller Orte und Personen (zwei Register!).
  - Auflistung der Vorkommen pro Entität (bspw. Sitzung 3 Eintrag 4; Sitzung 7 Eintrag 9; …)
  - Bonus: Integration und Verknüpfung mit Ergebnis von Übung 2

Vielen Dank für Ihre Aufmerksamkeit!

BERGISCHE
UNIVERSITÄT
WUPPERTAL

BERGISCHE
UNIVERSITÄT
WUPPERTAL