



UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

■ **Edirom Summer School 2010**

Workshop: XML und XML-Technologien

21.-24. November 2010

■ **M.Sc. Julian Dabbert**

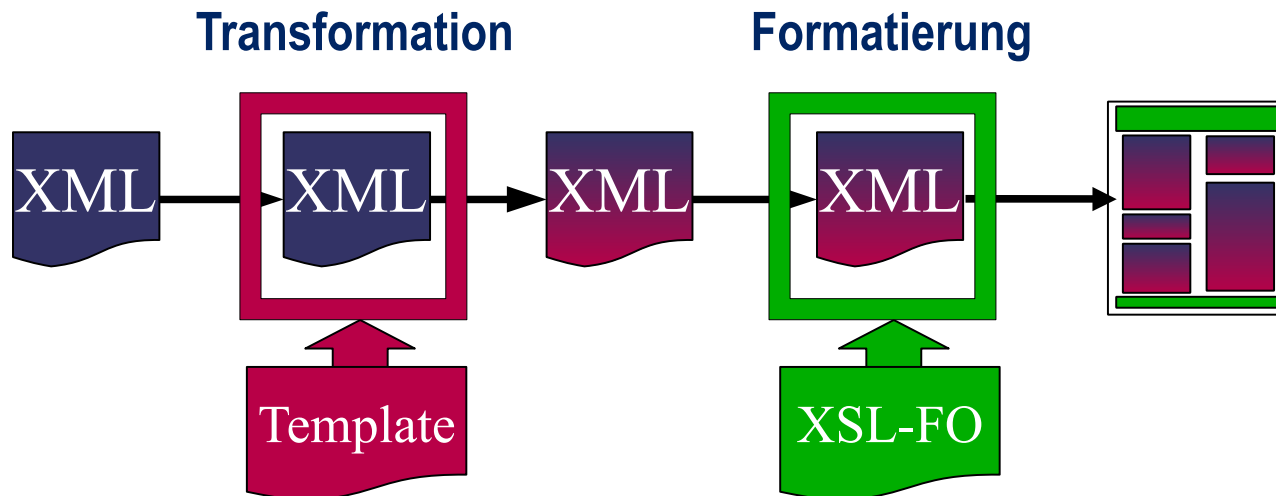
Fakultät für Kulturwissenschaften der Universität Paderborn

■ Block 6: Transformationen

■ Einführung XSL (Extensible Stylesheet Language)

Die Formatierungssprache XSL besteht aus zwei Teilen:

- XSL Transformationen (XSLT)
Transformation eines XML-Dokuments in ein anderes
- XSL Formatting Objects (XSL-FO)
Präsentation und Layout (ähnlich zu CSS)



■ Block 6: Transformationen

■ Parameter und Rückgabewert einer Transformation

Eine Transformation ist eine Funktion mit zwei Parametern:

- Das XML Dokument mit Daten
- Ein XSLT-Template (Stylesheet)

Diese Parameter werden zur Verarbeitung an einen XSLT-Prozessor übergeben, der beispielsweise in moderne Webbrowser integriert ist.

Der Rückgabewert der Transformation ist zumeist selbst wieder ein XML Dokument, das genau die Daten in genau der Formatierung enthält, die zur weiteren Verwendung benötigt werden

■ Block 6: Transformationen

■ Beispieldokument XML für Transformation (xslt_personen.xml)

```
<?xml version="1.0"?>
<people>
  <person born="1912" died="1954">
    <name>
      <first_name>Alan</first_name>
      <last_name>Turing</last_name>
    </name>
    <profession>computer
scientist</profession>
    <profession>mathematician</profession>
    <profession>cryptographer</profession>
  </person>
  <person born="1925" died="1986">
    <name>
      <first_name>Heinz</first_name>
      <last_name>Nixdorf</last_name>
    </name>
    <profession>computer pioneer</profession>
    <birthplace>Paderborn</birthplace>
  </person>
</people>
```

■ Block 6: Transformationen

■ XSL Template

Ein Template ist die Schablone, mit der das Quelldokument transformiert wird, um das Zieldokument zu erstellen

Beispiel: minimales XSLT Template

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

Ergebnis der Anwendung auf Beispieldokument:

```
<?xml version="1.0" encoding="utf-8"?>
```

Alan Turing
computer scientist mathematician cryptographer

Heinz Nixdorf
computer pioneer
Paderborn

■ Block 6: Transformationen

■ XSL Template Regeln – literale Ausgabe

Um die Ausgabe zu steuern, werden Regeln in das Stylesheet eingefügt.
Jede Regel wird durch ein `xsl:template` Element ausgedrückt

- Einfaches Suchmuster: Elementname

Beispiel: Für jedes Element namens „person“ gib aus: „Fund“

```
<xsl:template match="person">Fund</xsl:template>
```

Ergebnis:

```
<?xml version="1.0" encoding="utf-8"?>
```

Fund

Fund

■ Block 6: Transformationen

■ XSL Template Regeln – wertabhängige Ausgabe

Anstelle einer literalen Ausgabe kann auch der Wert eines Elements bearbeitet werden

- **Suchmuster: Elementname**

```
<xsl:template match="person">
  <p class = „note“>
    <xsl:value-of select="name"/>
  </p>
</xsl:template>
```

- **Ergebnis:**

```
<?xml version="1.0" encoding="utf-8"?>

<p class = „note“>
  Alan Turing
</p>

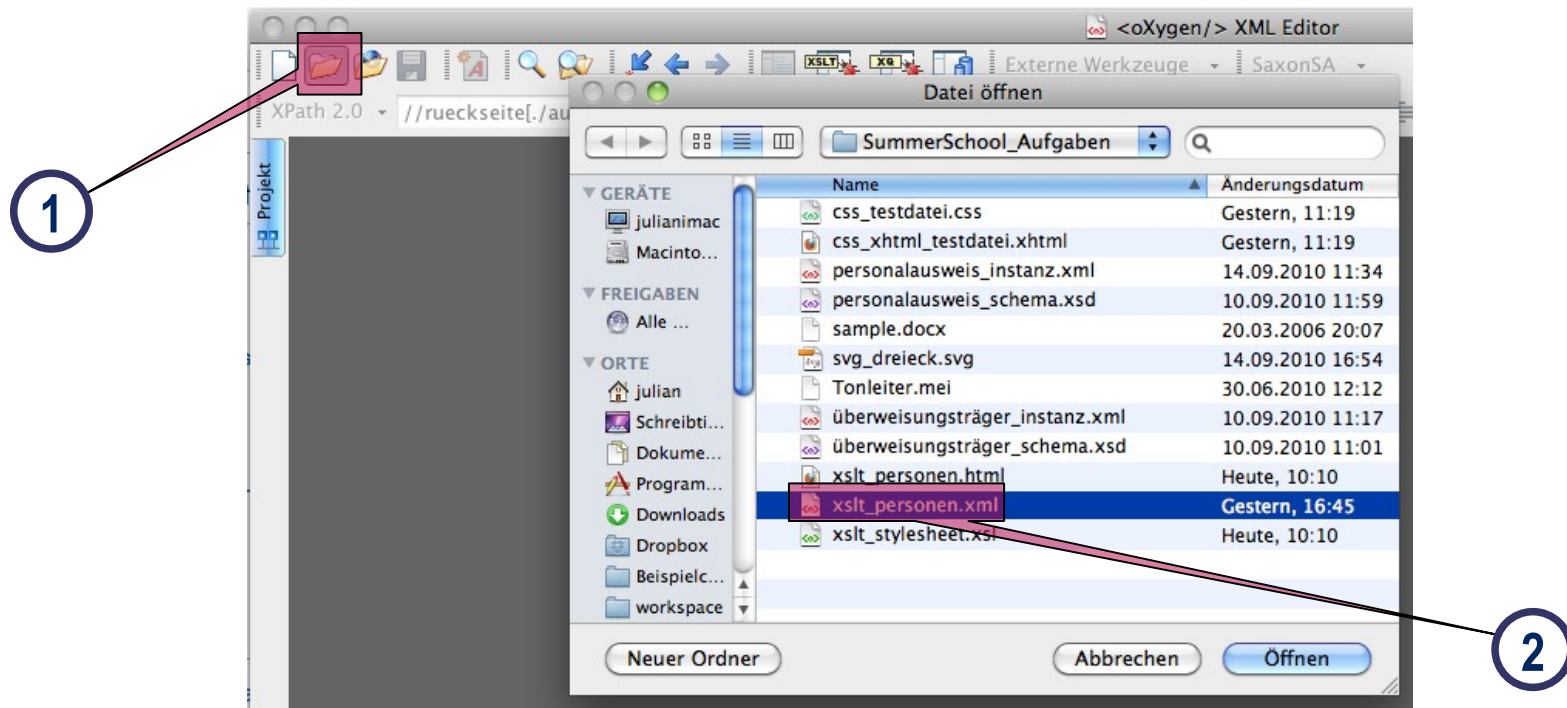
<p class = „note“>
  Heinz Nixdorf
</p>
```

■ Block 6: Transformationen

■ Übung: XSL Template

Führen Sie dieses Transformationsszenario mit dem oXygen Editor aus

- Schritt 1: Laden Sie die XML Datei „xslt_personen.xml“

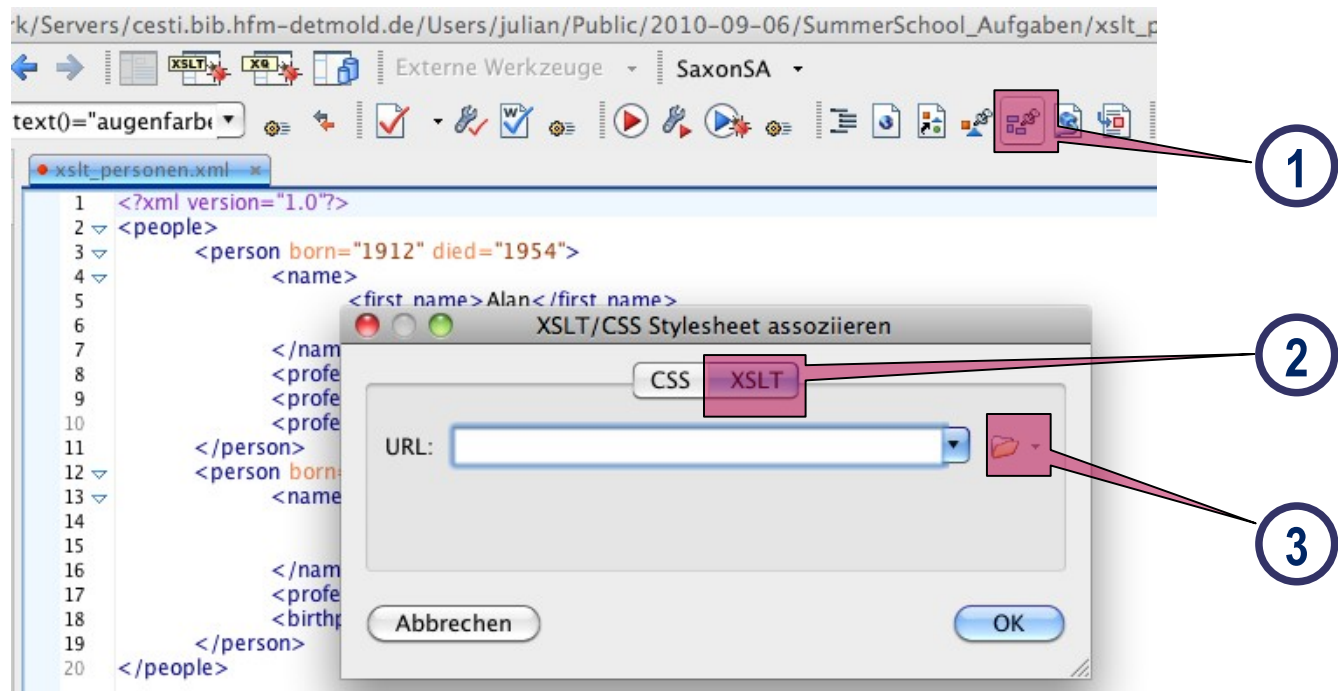


■ Block 6: Transformationen

■ Übung: XSL Template

Führen Sie dieses Transformationsszenario mit dem oXygen Editor aus

- Schritt 2: Assoziieren Sie ein Stylesheet

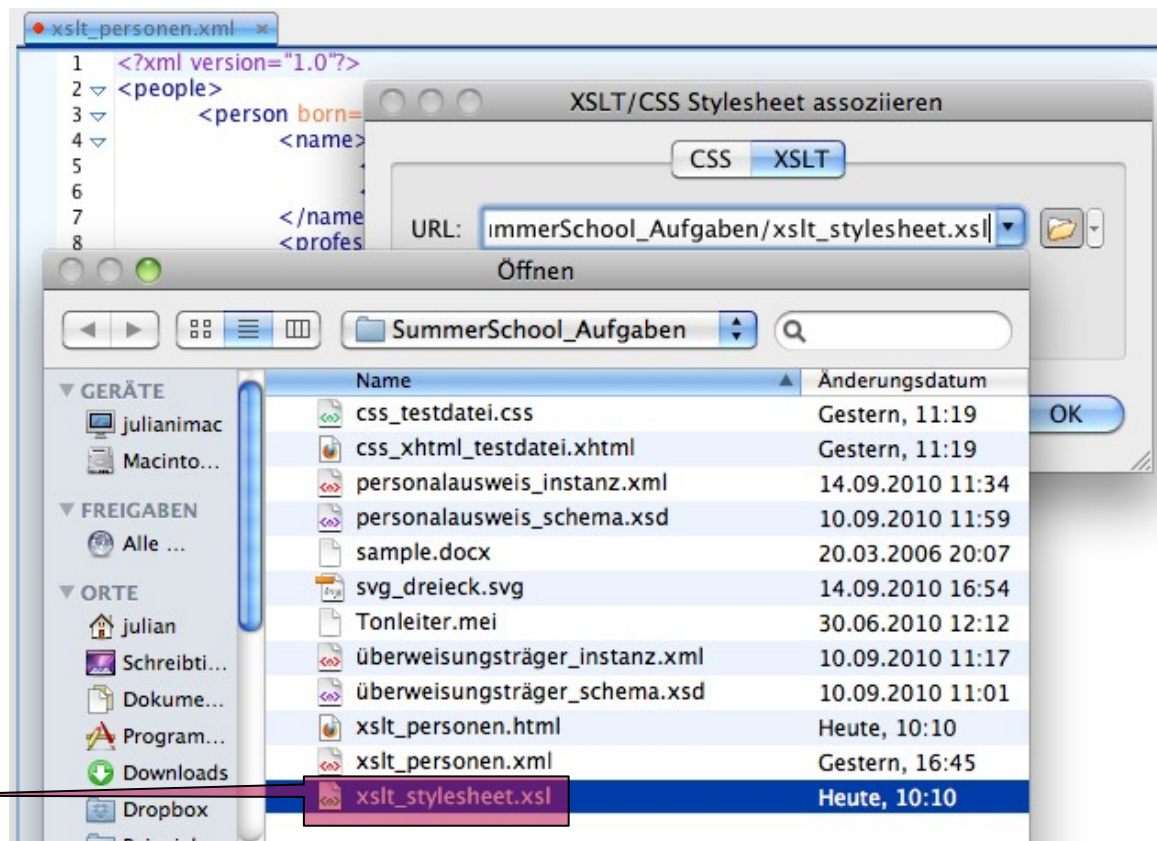


■ Block 6: Transformationen

■ Übung: XSL Template

Führen Sie dieses Transformationsszenario mit dem oXygen Editor aus

- Schritt 3: Wählen Sie das Stylesheet „xslt_stylesheet.xsl“ aus

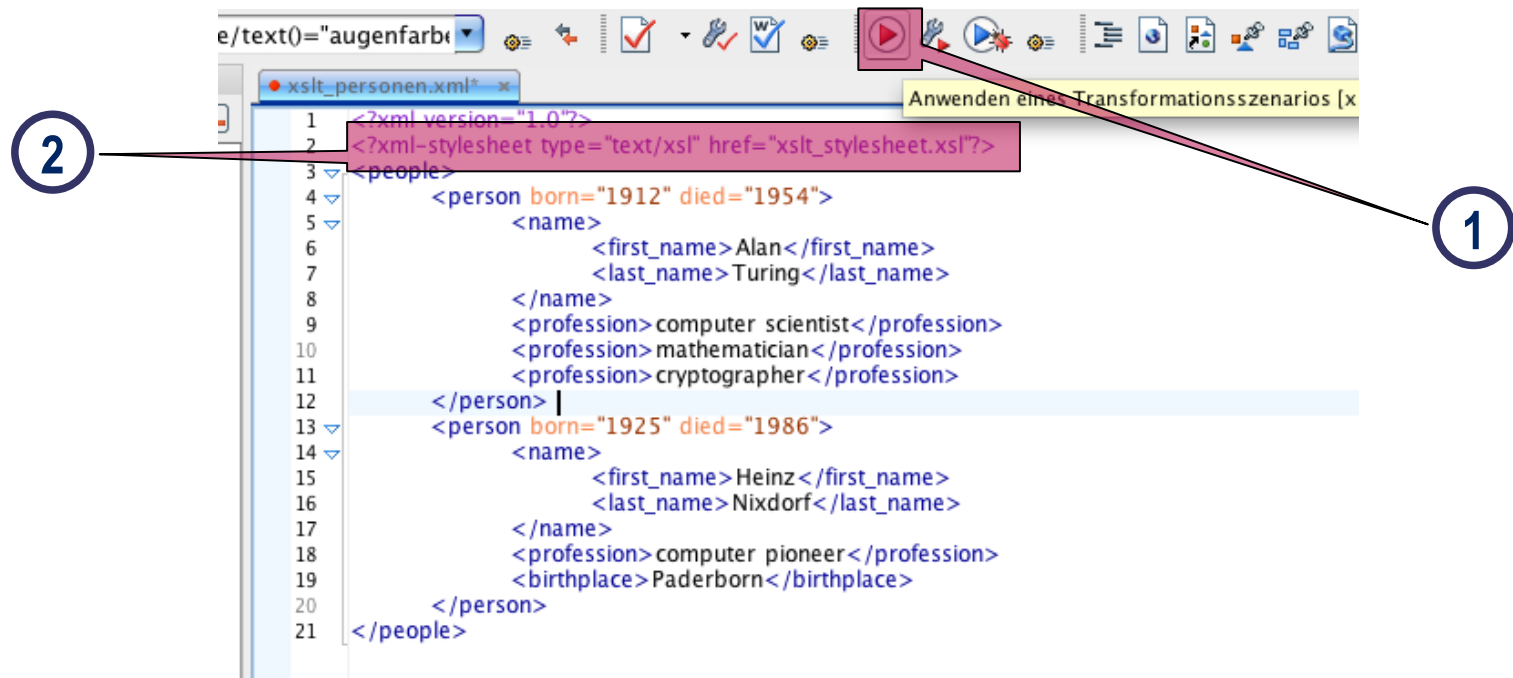


Block 6: Transformationen

Übung: XSL Template

Führen Sie dieses Transformationsszenario mit dem oXygen Editor aus

- Schritt 4: Führen Sie die Transformation durch



The screenshot shows the oXygen XML Editor interface. The toolbar at the top contains various icons, including a red play button icon labeled '1' with a red box around it, which is used to apply the transformation scenario. Below the toolbar, the editor window displays the XSL template code for 'xslt_personen.xml'. The code is as follows:

```
1 <?xml version="1.0"?>
2 <?xml-stylesheet type="text/xsl" href="xslt_stylesheet.xsl"?>
3 <people>
4   <person born="1912" died="1954">
5     <name>
6       <first_name>Alan</first_name>
7       <last_name>Turing</last_name>
8     </name>
9     <profession>computer scientist</profession>
10    <profession>mathematician</profession>
11    <profession>cryptographer</profession>
12  </person>
13  <person born="1925" died="1986">
14    <name>
15      <first_name>Heinz</first_name>
16      <last_name>Nixdorf</last_name>
17    </name>
18    <profession>computer pioneer</profession>
19    <birthplace>Paderborn</birthplace>
20  </person>
21 </people>
```

■ Block 6: Transformationen

■ Übung: XSL Template

Zusammenfassung Transformationsszenario mit oXygen:

- **Schritt 1: Laden Sie die XML Datei „xslt_personen.xml“**
- **Schritt 2: Assoziieren Sie ein Stylesheet**
- **Schritt 3: Wählen Sie das Stylesheet „xslt_stylesheetsheet.xsl“ aus**
- **Schritt 4: Führen Sie die Transformation durch**
- **Im Browser wird nun die Seite „xslt_personen.html“ angezeigt.
Diese Datei wurde durch die Transformation erstellt**

■ Block 6: Transformationen

■ XSL Template Regeln – Aufruf von Regeln

Die Reihenfolge der ausgeführten Regeln kann bestimmt werden, indem diese mittels `xsl:apply-templates` aufgerufen werden

- Suchmuster: Elementname, Template Regeln

```
<xsl:template match="name">
  <p class="note">
    <xsl:value-of select="./last_name"/>,
    <xsl:value-of select="first_name"/>
  </p>
</xsl:template>

<xsl:template match="person">
  <xsl:apply-templates select="name"/>
</xsl:template>
```

- Ergebnis: Ausgabe von Vorname, Nachname

■ Block 6: Transformationen

■ XSL Template Regeln – Einbettung in HTML

Die Ausgabe kann auch zielgerichtet in einen HTML Body geleitet werden. Dazu wird ein Rahmen eingeführt, in den die Ergebnisse der bisherigen Regeln eingebettet werden

- Suchmuster: Elementname, Template Regeln

```
<xsl:template match="people">
  <html>
    <head>
      <title>Famous Scientists</title>
    </head>
    <body>
      <xsl:apply-templates select="person"/>
    </body>
  </html>
</xsl:template>
```

- Ergebnis: Rahmen für HTML Datei

■ Block 6: Transformationen

■ Übung: XSL Template Regeln

Üben Sie die XSL Transformation mit der Verknüpfung eines Stylesheet.
Fügen Sie dazu die CSS Datei `css_testdatei.css` aus Block 5 als
Formatvorlage in das Stylesheet `xslt_stylesheet.xsl` ein

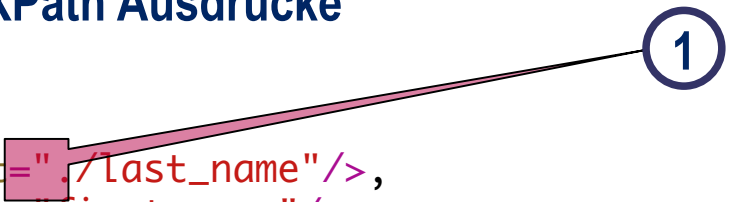
```
1 <xsl:template match="people">
    <html>
    <link [...]
    <head>
        <title>Famous Scientists</title>
    </head>
    <body>
        <xsl:apply-templates select="person"/>
    </body>
    </html>
</xsl:template>
2 <xsl:template [...]
```

■ Block 6: Transformationen

■ XSLT verwendet XPath

Im vorhin gezeigten XSLT war es bereits angedeutet:
XSLT Anweisungen enthalten XPath Ausdrücke

```
<xsl:template match="name">  
  <p class="note">  
    <xsl:value-of select="."/last_name"/>,  
    <xsl:value-of select="first_name"/>  
  </p>  
</xsl:template>
```



Diese können natürlich beliebig kompliziert werden:

```
<xsl:value-of select="//descendant::name[./last_name='Turing']"/>
```


■ Block 6: Transformationen

■ XSLT liest Attribute

Attribute werden in XSLT über XPath ausgelesen:

```
<xsl:template match="person">
  <xsl:apply-templates select="name">
    <p class="text">
      Born:<xsl:apply-templates select="@born"/>
      Died:<xsl:apply-templates select="@died"/>
    </p>
  </xsl:template>
```

■ Block 6: Transformationen

■ XSLT liest Werte in Variablen ein

Werte können eingelesen und an Variablen gebunden werden, die so wiederum für weitere Verwendung bereitstehen

```
<xsl:template match="name">
  <xsl:variable name="firstname">
    <xsl:apply-templates select="./child::first_name"/>
  </xsl:variable>
  <xsl:value-of select="$firstname"/>+
  <xsl:value-of select="$firstname"/>+
  <xsl:value-of select="$firstname"/>
</xsl:template>
```

①

②

■ Block 6: Transformationen

■ XSLT Anwendungsbeispiel

Weil XSLT gut auf XML Daten operiert, ist es besonders nützlich für die Konvertierung von XML Dokumenten von einem Format in ein anderes.

Beispiel aus dem MuWi: MusicXML → MEI

- Daten aus Quelldatei in MusicXML werden eingelesen und in das Zielformat MEI umgeformt
- MEI kann das ausdrücken, was MusicXML beschreibt, daher kann die Umwandlung ohne Verlust von Informationen stattfinden



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft



Ende des Blocks 6