Coding the Sources

Digitales Edieren in den Geisteswissenschaften

Online Summerschool und Workshop

Lübeck/Online, 17.08. – 27.08.2020

Forschungsstelle
für die Geschichte
der Hanse und des Ostseeraums

i|d|e

Christian-Albrechts-Universität zu Kiel
Historisches Seminar

Einführung in XSLT

Patrick Sahle
sahle@uni-wuppertal.de
Lübeck/Online, 25.08.2020

Forschungsstelle
für die Geschichte
der Hanse und des Ostseeraums

Christian-Albrechts-Universität zu Kiel
Historisches Seminar

Übersicht

- Was ist XSLT?
- Grundsätzliche Funktionsweise
- Aufbau einer Transformation
- Wichtige ~~Befehle~~ Elemente
- XSLT in oXygen
- Übungen

Was ist XSLT?

- XSLT ist eine Sprache, um XML-Dokumente in verschiedene Zielformate zu transformieren
- XSLT ist ein W3C-Standard seit 1999, aktuell: XSLT 3.0
- *XSLT bedeutet*
  Extensible Stylesheet Language Transformation
- XSLT ist selbst XML
- XSL = XSLT + XSL-FO

Wozu braucht man XSLT?

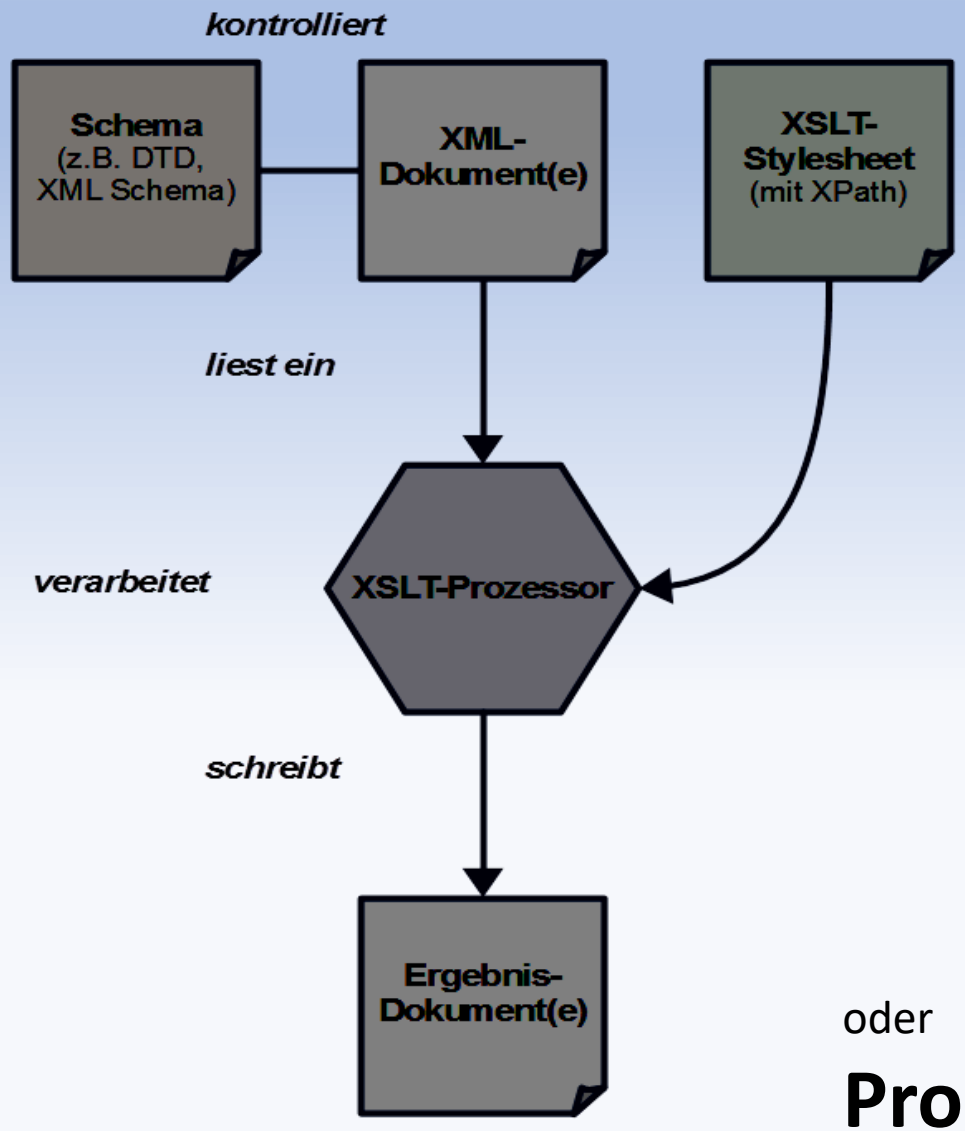
Man braucht XSLT, um aus XML etwas (anderes) zu machen

- Reports → Z.B. Text
- Publikationen → Z.B. HTML/CSS
- Visualisierungen → Z.B. SVG, HTML/CSS/Javascript
- Exportformate → Z.B. CSV, JSON, Datenbankformate
- XML → Daten verändern, anreichern, zusammenführen, extrahieren, konvertieren …

… dies alles kann zusammenspielen!

… es gibt praktisch nichts, was man nicht erzeugen könnte

Grundsätzliche Funktionsweise



- XSLT verwendet XPath
- XSLT wird von einem XSLT-Prozessor ausgeführt
- XML + XSLT
  → Zieldokument(e)

oder

Prozessor(XML+XSLT(XPath)) → Zieldokument

Grundsätzliche Funktionsweise

Das Prinzip der drei Bäume:

- Eingabebaum – Verarbeitungsbaum – Ausgabebaum

Das Prinzip der Verarbeitung

- Gehe durch den Eingabebaum und berücksichtige dabei die Anweisungen des Verarbeitungsbaums (seine Schablonen) um den Ausgabebaum zu schreiben

Das Prinzip der „Schablonen“

- Wenn diese Schablone (ein XPath-Muster) passt, dann folge ihren Anweisungen

Grundsätzliche Funktionsweise

Deutsch → XSLT

Ich bin ein XML-Dokument

Ich bin ein XSLT-Stylesheet

Ich bin eine Schablone, ich passe auf ein Muster

Ich tue etwas, wenn das Muster im Ausgangsbaum gefunden wird. Meistens hole ich Informationen aus dem XML-Dokument, verarbeite sie und schreibe sie in das Zieldokument

Ich bin eine weitere Schablone, ich passe auf ein Muster …

Grundsätzliche Funktionsweise

Deutsch → XSLT

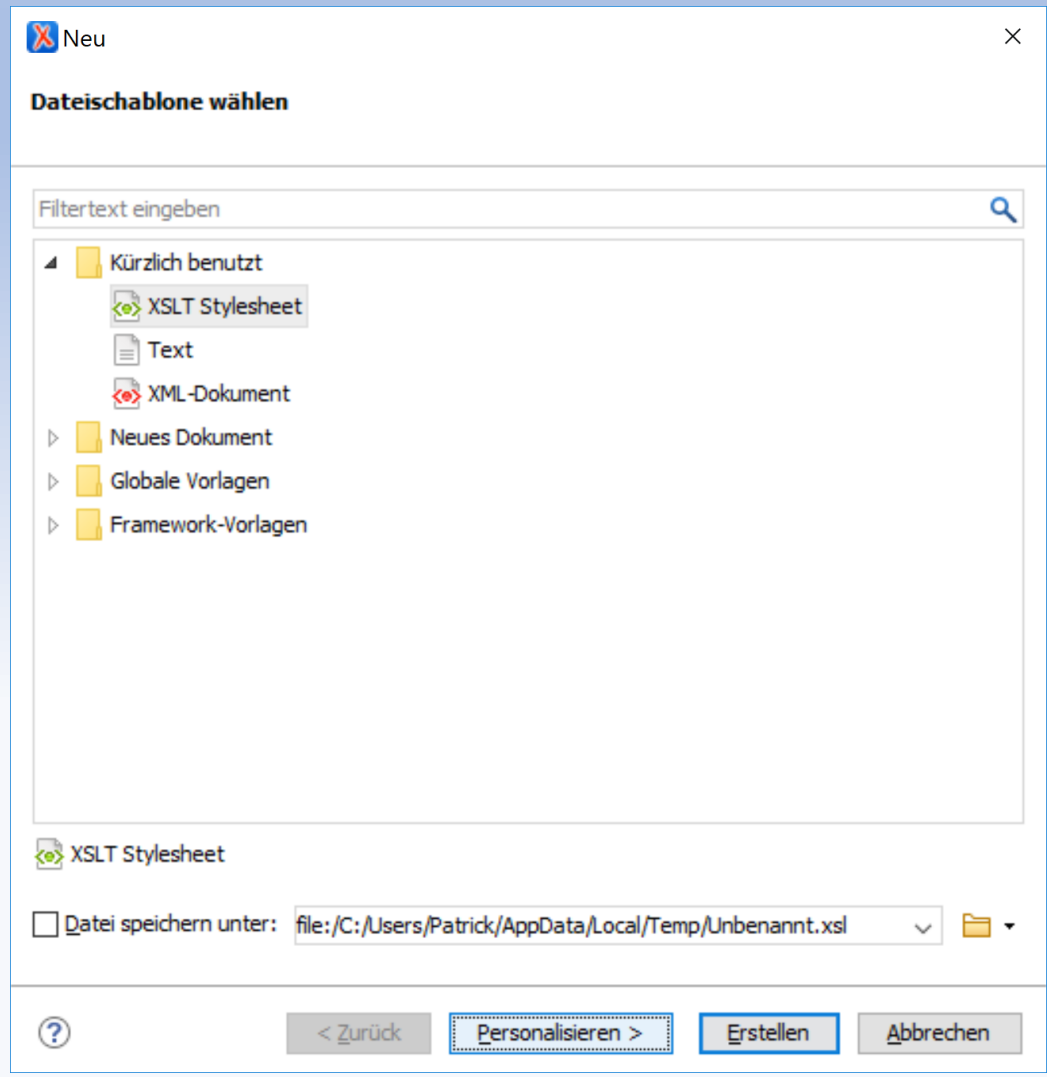
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform
  version="2.0">
    <xsl:template match="/">
        <h1>
            <xsl:value-of select="//titel" />
        </h1>
        <xsl:text>Hallo Welt</xsl:text>
    </xsl:template>
    <xsl:template match="lb"><br/></xsl:template>
</xsl:stylesheet>
```

Ein XSLT-Stylesheet in oXygen

- Strg+n

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

</xsl:stylesheet>
```

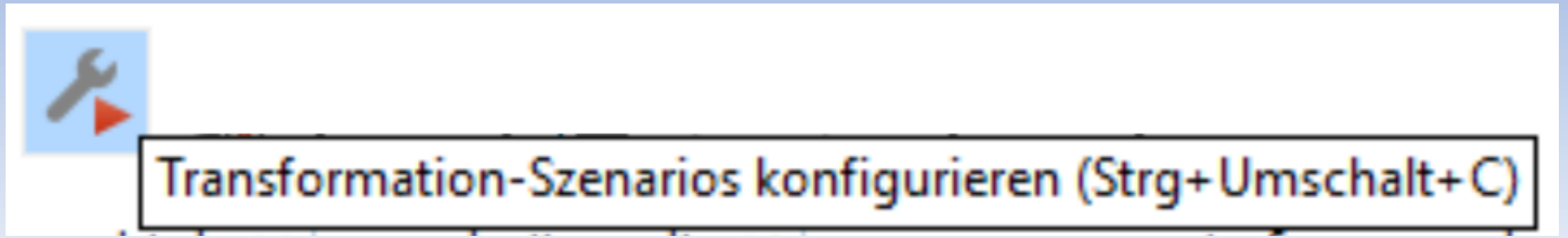


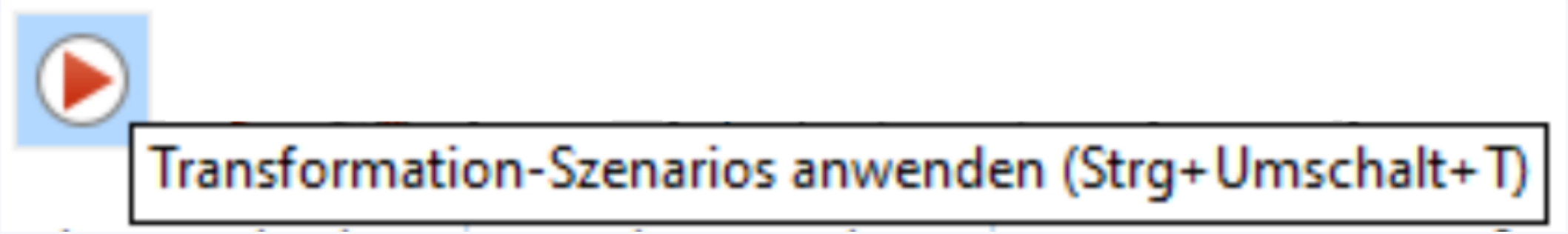
Transformationsszenarien

Wie bringt man XML, XSLT und den Prozessor zusammen?

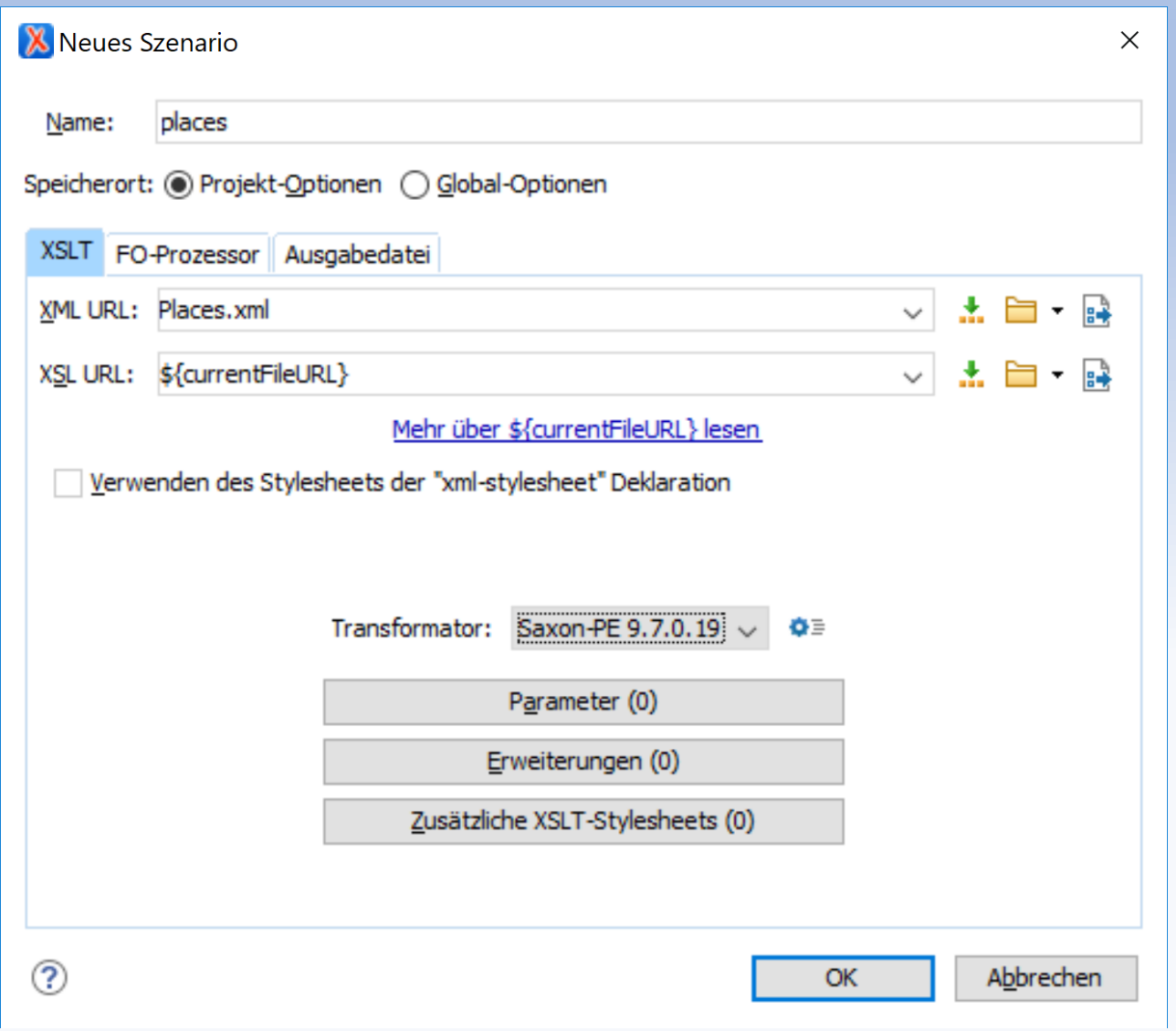
a) Dieses XML-Dokument soll mit jenem XSLT-Dokument verarbeitet werden (processing instruction)
b) Lieber Prozessor, verarbeite jenes XML-Dokument mit jenem XSLT-Dokument
   → oXygen „Transformationsszenario“

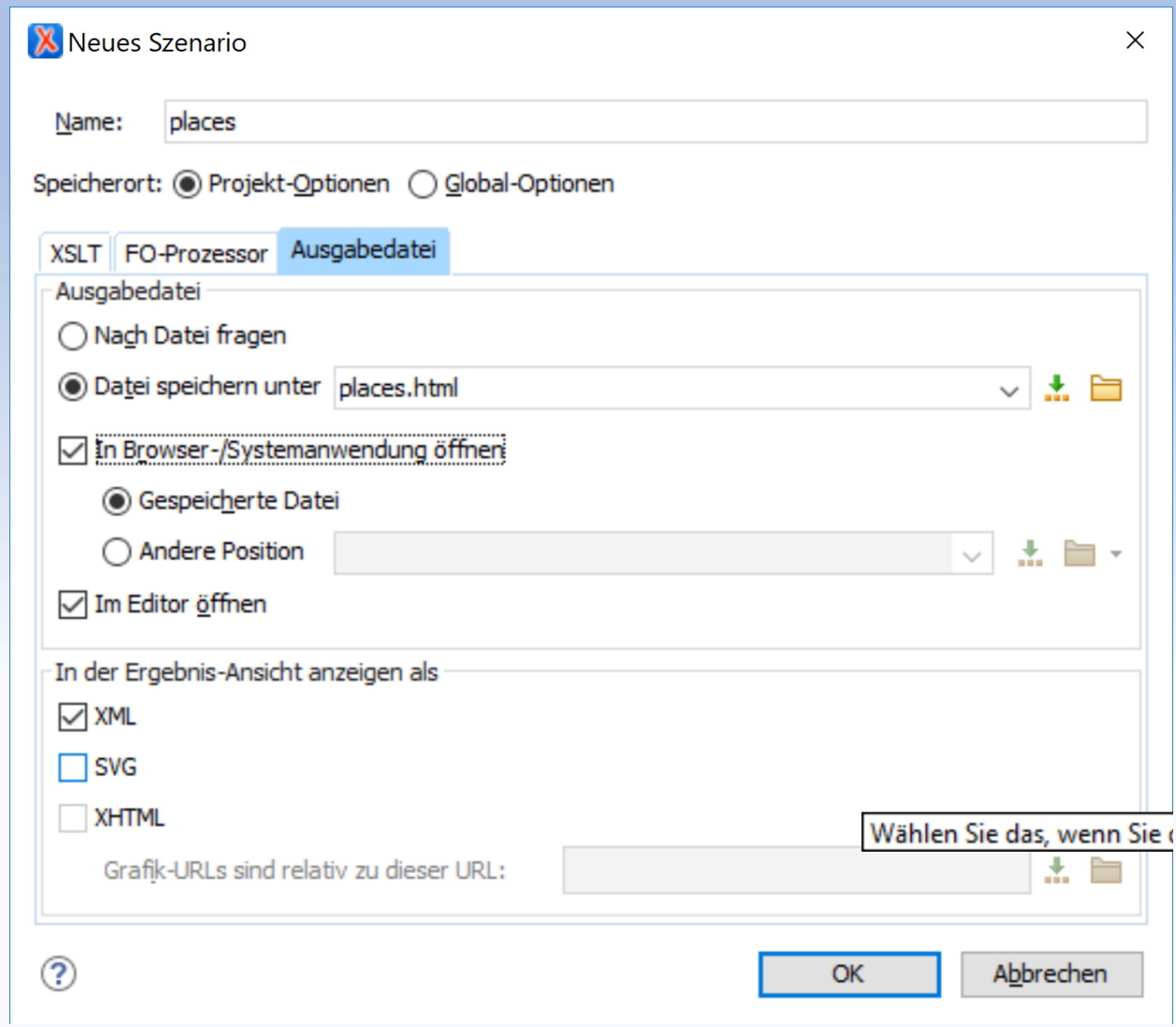
oXygen-Transformationsszenario





oXygen-Transformationsszenario





Einige ~~Befehle~~ Elemente

Vorab:

- Nochmal zu XSLT 1.0, XSLT 2.0, XSLT 3.0 …
- Ca. 50 Elemente (in XSLT 2.0)
- Von denen Sie ca. 10-20 brauchen (um schon mal sehr weit zu kommen)
  - Erklärung: ein großer Teil der Magie findet in XPath statt
- Einfaches Tutorial (etwas veraltet): https://www.w3schools.com/xml/xsl_intro.asp
- Nicht gut (nur XSLT 1.0): https://www.w3schools.com/xml/xsl_elementref.asp
- Eine etwas bessere Referenz (mit XSLT 2.0): https://www.data2type.de/xml-xslt-xslfo/xslt/xslt-referenz/alphabetische-referenz-der-xslt-elemente/

Einige Elemente

xsl:stylesheet	ist das Wurzelelement eines XSL-Dokumentes.
xsl:template	enthält Instruktionen, deren Ausgabe direkt in das Ergebnisdokument eingetragen werden. Das Attribut match(*XPath*) sagt, auf welches Muster die Schablone passen soll
xsl:apply-templates	ruft das Template auf, dessen `match`-Attribut auf ein Kindknoten des aktuellen Kontextknotens passt.
xsl:value-of	wandelt eine im `select`-Attribut angegebene Sequenz in Werte um, die als Output in das Ergebnisdokument kopiert werden.

Einige Elemente

<xsl:value-of select="XPath-Ausdruck">

1. value-of ist der zentrale „Ausgabebefehl“
2. Es gibt Situationen, in denen man sich schon innerhalb von XML-Ausgabekonstrukten befindet. Dann gibt es eine stark verkürzte Syntax
   …
   Beispiel: <a href="{Xpath-Ausdruck}">

Einige Elemente

xsl:for-each	Tue für jedes … etwas. Menge wird im Attribut select bestimmt.
xsl:for-each-group	dient dazu, eine im `select`-Attribut angegebene Eingabesequenz anhand eines Kriteriums (@group-by) zu gruppieren. Man hat dann Zugriff auf die Gruppen mit current-group() und auf den Gruppierungsschlüssel mit current-grouping-key()
xsl:sort	sortiert eine (Knoten-)Sequenz anhand eines durch das `select`-Attribut ausgewählten Sortierkriteriums.
xsl:if	die klassische IF-Abfrage, ohne jedoch eine alternative ELSE-Möglichkeit zuzulassen.
xsl:choose	enthält eine oder mehrere `xsl:when`- und gegebenenfalls abschließend eine `xsl:otherwise`-Instruktion.
xsl:variable	definiert eine Variable, die mit einem Namen als Referenz versehen wird und der Werte, Strings, Elemente oder ganze Teilbäume zugeordnet werden können.

Zwei Strategien

Pull

- Hole gezielt Daten und tue etwas damit
  → typisch: for-each, for-each-group

Push

- Verarbeite Daten, wenn sie Dir über den Weg laufen
  → typisch: template, apply-templates, call-template

Wenn etwas schief läuft

- Die Fehlermeldungen sind meistens sehr aussagekräftig!
- Wenn zuviel rauskommt: das unsichtbare template für text()
- Wenn nichts rauskommt ... Typische Ursachen …
  - Der Kontext ist ein anderer, als man denkt
  - Namespaces!!!
- Debugging
  - Adler und Robben
  - XSLT Debugger in oXygen
  - Kontrollausgaben

Übungen

1. Hallo Welt
2. Register erzeugen
3. Transkription ausgeben

Zur Orientierung (Zielaussehen, Zielsyntax)
siehe uebungN-n.html in google drive

Übung 1: Hallo Welt

1. Ein XSLT-File anlegen
2. XSLT schreiben
    1. Ein Stylesheet-Element anlegen. Als Version nehmen wir mal 2.0
    2. Man braucht immer ein template! Für den Anfang match="/"
    3. Was soll ausgegeben werden? Hallo Welt!
    4. Man könnte das schöner machen, als HTML. Mit <html><head></head><body><h1>…
3. Ein Verarbeitungsszenario anlegen
    1. Mit dem rezess.xml verbinden
    2. Ausgabedatei festlegen (z.B. hallo.html), in Browser öffnen lassen
4. Press Play

HTML? CSS?

- Auch wenn wir nur Ausgaben für uns selbst erzeugen, kann HTML/CSS sinnvoll sein (Darstellung im Browser)
- HTML = Seitenstrukturierungssprache für das Internet
- CSS = Cascading Style Sheets, „Schönmachsprache“
- Wie lernt man das nötigste?
  - Tutorials HTML CSS
  - Rechte Maustaste auf Webseiten: „Untersuchen“ o.ä.; Seitenquelltext ansehen
  - Googlen …

Übung 2: Register erzeugen

1. Ein XSLT-File anlegen und unter einem Namen speichern, an einem Ort, den man wiederfindet …
2. XSLT schreiben
    1. Ein stylesheet anlegen, wie gehabt
    2. Achtung: das XML-File hat einen Namensraum! Damit sich XML und XSLT blind verstehen, bitte im Stylesheet-Element eintragen: xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    3. Man braucht immer ein template! Für den Anfang match="/"
    4. Einen selbst erfundenen Seitentitel ausgeben lassen, z.B. „Personenregister“, in html schön machen?
    5. Die Personen aus dem XML holen und ausgeben
        1. Wir brauchen eine Schleife (deutsch: für jedes …), die die Personen ansteuert (XPath?)
        2. Wir geben die Namen der jeweiligen Person aus (deutsch: Wert von)
        3. Und erzeugen noch einen Umbruch hinter der Person

Übung 2: Register erzeugen

3. Ein Verarbeitungsszenario anlegen
   1. Mit dem rezess.xml verbinden
   2. Ausgabedatei festlegen (z.B. register.html), in Browser öffnen lassen
4. Press Play und staune. Das Ergebnis sollte aussehen wie uebung2-a.html in drive
5. Dem Hündchen neue Tricks beibringen
   1. [z.B. Ausgabe schöner machen (HTML+CSS) …]
   2. Eine explizite HTML-Liste? (Suche HTML unsorted list)
   3. Die Liste sortieren (deutsch: sortieren), Achtung: der XSL-Befehl kommt direkt hinter dem Schleifen-Element! Wonach soll sortiert werden? Z.B. nach dem Nachnamen? Zum Spaß: Alphabetisch abwärts
   4. Mehr Angaben aus dem XML holen? (Berufe, Rollen, Lebensdaten, die GND-Links als HTML-Links? Auf Text „GND“ oder Logo?)

Übung 2: Register erzeugen

5. Dem Hündchen neue Tricks beibringen
    5. Das Register in Buchstabenblöcke unterteilen, die jeweils den Buchstaben vorangestellt haben.
        1. Dazu brauchen wir ein neues Element, deutsch: für jede Gruppe. Das nimmt wieder alle Personen und gruppiert sie. Überlegen: wonach gruppieren wir?
        2. Sortieren lassen (zum Spaß: absteigend)
        3. Für jede Gruppe soll der jeweilige Buchstabe (deutsch: aktueller Gruppierungsschlüssel) ausgegeben werden, z.B. in <h4>
        4. Unterschleife (deutsch: für jedes) für die Mitglieder der aktuellen Gruppe (deutsch: aktuelle Gruppe)
        5. Infos zur Person ausgeben lassen
    6. Ein Ortsregister anlegen
        1. Wie bei den Personen …
        2. Man könnte Gruppen bilden nach dem Ortstyp
        3. Man könnte die Orts-Identifier als Hyperlinks benutzen (in HTML <a href="link">)

Übung 3: Transkription ausgeben lassen

1. Ein XSLT-File anlegen
2. XSLT schreiben
    1. Achtung: das XML-File hat einen Namensraum! Damit sich XML und XSLT blind verstehen, bitte im Stylesheet-Element eintragen: xpath-default-namespace="http://www.tei-c.org/ns/1.0"
    2. Man braucht immer ein template! Für den Anfang match="/"
    3. Wir holen uns nicht die Infos selbst, sondern übergeben die einfach weiteren templates: im Grundtemplate (deutsch: wende Schablonen an – ein leeres Element)
    4. Zwei leere Schablonen für teiHeader und standOff, damit die nicht ausgegeben werden

Übung 3: Transkription ausgeben lassen

2. XSLT schreiben
    5. Eine Schablone für die Seitenumbrüche. Dabei ein <h2> mit der Nummer der Seite ausgeben lassen
    6. Eine Schablone für die Textabsätze <p>, die in HL als <p> ausgegeben werden sollen
    7. Eine Schablone für die Zeilenumbrüche. Wie heißen Zeilenumbrüche bei HTML?
    8. Das Ergebnis bis hier liegt in Drive als uebung3-a.html
3. Das Script weiter ausbauen
    1. Eine Schablone für die Ortsnamen, so dass die Ortsnamen in rot und fett ausgegeben werden. Z.B. mit dem HTML-Element span und seinem Attribut style und dessen CSS-Angaben für Schriftgewicht und Farbe. Die Auflösung der Daten als tool-tip (@title)
    2. Was ist mit <hi>?

Fragen?

sahle@uni-wuppertal.de

Forschungsstelle
für die Geschichte
der Hanse und des Ostseeraums

i|d|e

Christian-Albrechts-Universität zu Kiel
Historisches Seminar