Einführung in XPath

Schulung für das

Editionsprogramm Fraktionen im Deutschen Bundestag 1949 – 1990

04./05.10.2021

Übersicht

- XPath als Standard und Hilfsmittel
- Konzepte und Funktionsweise
- Zusatz: Regex
- Übungen

Was ist XPath?

- „XPath is a language for addressing parts of an XML document” (XPath 1.0 W3C Specification)
- XPath dient
  - der Navigation in XML-Dokumenten,
  - der Manipulation von Strings, Zahlen und Wahr/Falsch-Aussagen (Booleans),
  - der Erzeugung von “Rückgaben”
- XPath wird vor allem in anderen X-Technologien verwandt: XSLT, Xquery („designed to be embedded in a host language”)
- Vor allem notwendig, um XML-Dokumente zu verstehen und zu „befragen“

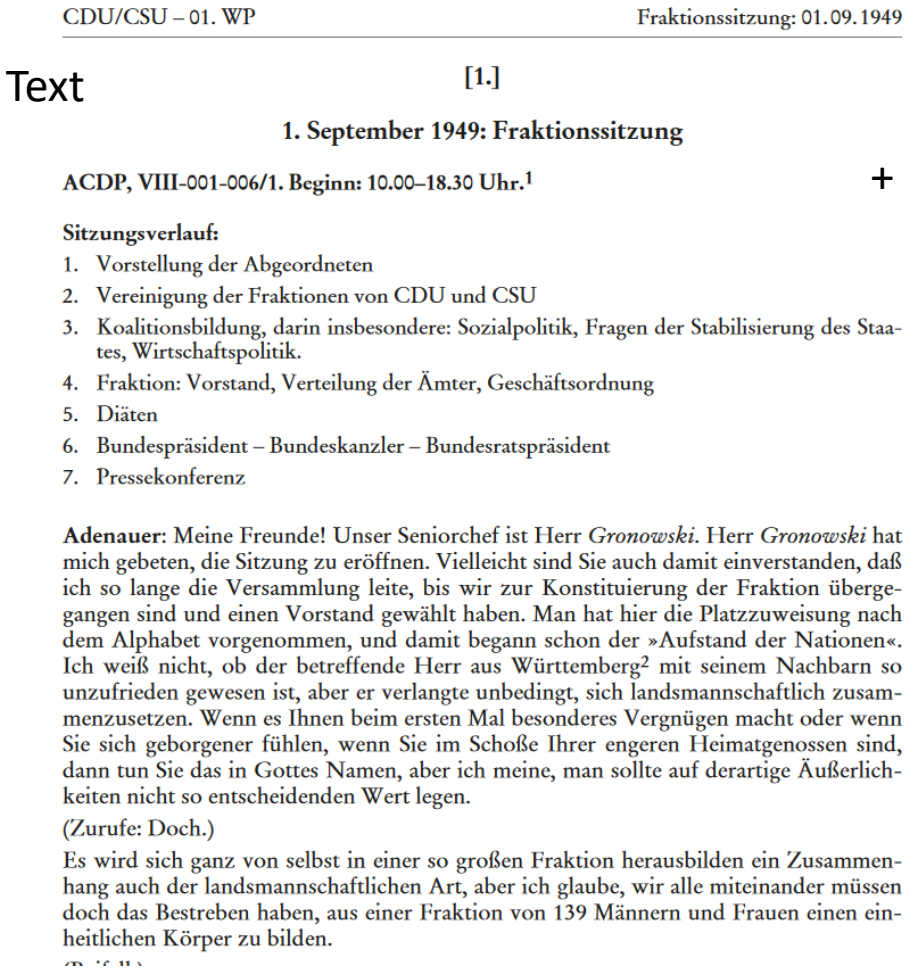
Standard

- Xpath ist ein W3C-Standard
  - Version 1.0–1999: https://www.w3.org/TR/1999/REC-xpath-19991116/
  - Version 2.0–2010
  - Version 3.1–2017: https://www.w3.org/TR/xpath-3/
- Für Selbstlerner*innen: XPath Tutorial des W3C https://www.w3schools.com/xml/xpath_intro.asp

Der Wald vor lauter Bäumen…

Auszeichnungen im Autormodus

Text



CDU/CSU – 01. WP Fraktionssitzung: 01.09.1949

[1.]

1. September 1949: Fraktionssitzung

ACDP, VIII-001-006/1. Beginn: 10.00–18.30 Uhr.[1]

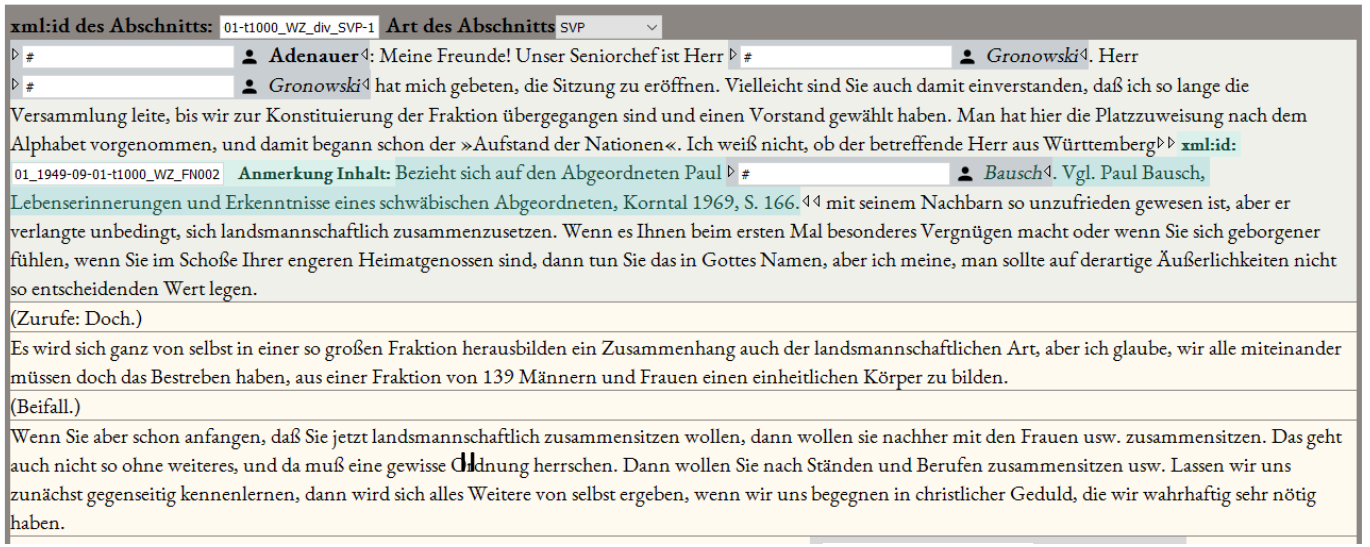
Sitzungsverlauf:

1. Vorstellung der Abgeordneten
2. Vereinigung der Fraktionen von CDU und CSU
3. Koalitionsbildung, darin insbesondere: Sozialpolitik, Fragen der Stabilisierung des Staates, Wirtschaftspolitik.
4. Fraktion: Vorstand, Verteilung der Ämter, Geschäftsordnung
5. Diäten
6. Bundespräsident – Bundeskanzler – Bundesratspräsident
7. Pressekonferenz

Adenauer: Meine Freunde! Unser Seniorchef ist Herr *Gronowski*. Herr *Gronowski* hat mich gebeten, die Sitzung zu eröffnen. Vielleicht sind Sie auch damit einverstanden, daß ich so lange die Versammlung leite, bis wir zur Konstituierung der Fraktion übergegangen sind und einen Vorstand gewählt haben. Man hat hier die Platzzuweisung nach dem Alphabet vorgenommen, und damit begann schon der »Aufstand der Nationen«. Ich weiß nicht, ob der betreffende Herr aus Württemberg[2] mit seinem Nachbarn so unzufrieden gewesen ist, aber er verlangte unbedingt, sich landsmannschaftlich zusammenzusetzen. Wenn es Ihnen beim ersten Mal besonderes Vergnügen macht oder wenn Sie sich geborgener fühlen, wenn Sie im Schoße Ihrer engeren Heimatgenossen sind, dann tun Sie das in Gottes Namen, aber ich meine, man sollte auf derartige Äußerlichkeiten nicht so entscheidenden Wert legen.

(Zurufe: Doch.)

Es wird sich ganz von selbst in einer so großen Fraktion herausbilden ein Zusammenhang auch der landsmannschaftlichen Art, aber ich glaube, wir alle miteinander müssen doch das Bestreben haben, aus einer Fraktion von 139 Männern und Frauen einen einheitlichen Körper zu bilden.



xml:id des Abschnitts: 01-t1000_WZ_div_SVP-1 **Art des Abschnitts** SVP

Adenauer: Meine Freunde! Unser Seniorchef ist Herr *Gronowski*. Herr *Gronowski* hat mich gebeten, die Sitzung zu eröffnen. Vielleicht sind Sie auch damit einverstanden, daß ich so lange die Versammlung leite, bis wir zur Konstituierung der Fraktion übergegangen sind und einen Vorstand gewählt haben. Man hat hier die Platzzuweisung nach dem Alphabet vorgenommen, und damit begann schon der »Aufstand der Nationen«. Ich weiß nicht, ob der betreffende Herr aus Württemberg **xml:id:** 01_1949-09-01-t1000_WZ_FN002 **Anmerkung Inhalt:** Bezieht sich auf den Abgeordneten Paul *Bausch*. Vgl. Paul Bausch, Lebenserinnerungen und Erkenntnisse eines schwäbischen Abgeordneten, Korntal 1969, S. 166. mit seinem Nachbarn so unzufrieden gewesen ist, aber er verlangte unbedingt, sich landsmannschaftlich zusammenzusetzen. Wenn es Ihnen beim ersten Mal besonderes Vergnügen macht oder wenn Sie sich geborgener fühlen, wenn Sie im Schoße Ihrer engeren Heimatgenossen sind, dann tun Sie das in Gottes Namen, aber ich meine, man sollte auf derartige Äußerlichkeiten nicht so entscheidenden Wert legen.

(Zurufe: Doch.)

Es wird sich ganz von selbst in einer so großen Fraktion herausbilden ein Zusammenhang auch der landsmannschaftlichen Art, aber ich glaube, wir alle miteinander müssen doch das Bestreben haben, aus einer Fraktion von 139 Männern und Frauen einen einheitlichen Körper zu bilden.

(Beifall.)

Wenn Sie aber schon anfangen, daß Sie jetzt landsmannschaftlich zusammensitzen wollen, dann wollen sie nachher mit den Frauen usw. zusammensitzen. Das geht auch nicht so ohne weiteres, und da muß eine gewisse Ordnung herrschen. Dann wollen Sie nach Ständen und Berufen zusammensitzen usw. Lassen wir uns zunächst gegenseitig kennenlernen, dann wird sich alles Weitere von selbst ergeben, wenn wir uns begegnen in christlicher Geduld, die wir wahrhaftig sehr nötig haben.

XML-„Baum“

```
<body>
    <div type="SVP" xml:id="cdu-csu-01_1949-09-01-t1000_WZ_div_SVP-1">
        <p><name type="Person" role="Sprecher" ref="#">Adenauer</name>: Meine Freunde! Unser Seniorchef ist Herr
        <p>(Zurufe: Doch.)</p>
        <p>Es wird sich ganz von selbst in einer so großen Fraktion herausbilden ein Zusammenhang auch der landsmanns
        <p>(Beifall.)</p>
        <p>Wenn Sie aber schon anfangen, daß Sie jetzt landsmannschaftlich zusammensitzen wollen, dann wollen sie nac
        <p>Nun sind sehr viele unter uns, die sich nicht gegenseitig kennen. Ich schlage deshalb vor, daß Herr <name type
        <p><name type="Person" role="Sprecher" ref="#">Dörpinghaus</name> verliest die Namen sämtlicher Mitglieder
        <p><name type="Person" role="Sprecher" ref="#">Adenauer</name>: Ich erteile nunmehr das Wort Herrn Staatsı
        <p>Staatsrat <name type="Person" role="Sprecher" ref="#">Schäffer</name>: Meine sehr verehrten Damen und
```

Der Wald vor lauter Bäumen…

- Text als **Sequenz**… Meine Freunde! Unser Seniorchef ist Herr Gronowski. Herr

- … wird mit **tags** umgeben -> Text + tags (+Attribute) = **Elemente**

Meine Freunde! Unser Seniorchef ist Herr <name type="Person" role="Erwaehnung" ref="#">Gronowski</name>. Herr <name type="Person" role="Erwaehnung" ref="#">G

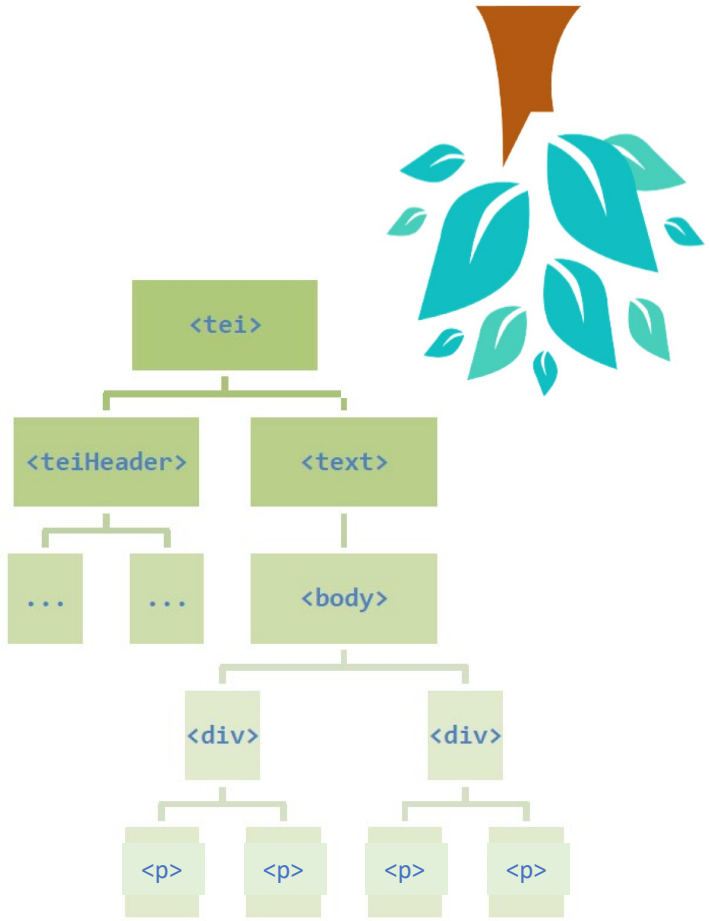
- … Elemente werden von Elementen umgeben -> **Verschachtelung**

```
<div type="SVP" xml:id="cdu-csu-01_1949-09-01-t1000_WZ_div_SVP-1">
    <p>
        <name type="Person" role="Sprecher" ref="#">Adenauer</name>: Meine Freunde! Unser Seniorchef ist Herr Gronowski. Herr
        <name type="Person" role="Erwaehnung" ref="#">Gronowski</name> hat mich gebeten, die Sitzung zu eröffnen. Vielleicht sind
        <seg type="note">
            <note xml:id="cdu-csu-01_1949-09-01-t1000_WZ_FN002" type="comment">
                <p>Bezieht sich auf den Abgeordneten Paul
                    <name type="Person" role="Erwaehnung" ref="#">Bausch</name>. Vgl. Paul Bausch, Lebenserinnerunge
                </p>
            </note>
        </seg> mit seinem Nachbarn so unzufrieden gewesen ist, aber er verlangte unbedingt, sich landsmannschaftlich zusammenzuset
    </p>
```

Der Wald vor lauter Bäumen…

- Ein „allererstes“ Element umschließt das gesamte Dokument
  -> Wurzel-Element
  -> Hierarchie von oben nach unten
  -> Ein „umgedrehter Baum“?

```
<TEI xmlns="http://www.tei-c.org/ns/1.0" xml:id="cdu-csu-01_1949-09-01-t1000_WZ" rendition="fraktionsprotokolle">
    <teiHeader> [115 lines]
    <text>
        <front> [17 lines]
        <body>
            <div type="SVP" xml:id="cdu-csu-01_1949-09-01-t1000_WZ_div_SVP-1">
                <p>
                    <name type="Person" role="Sprecher" ref="#">Adenauer</name>: Meine Freunde! Unser Seniorchef is
                    <name type="Person" role="Erwaehnung" ref="#">Gronowski</name> hat mich gebeten, die Sitzung zu
                    <seg type="note">
                        <note xml:id="cdu-csu-01_1949-09-01-t1000_WZ_FN002" type="comment">
                            <p>Bezieht sich auf den Abgeordneten Paul
                                <name type="Person" role="Erwaehnung" ref="#">Bausch</name>. Vgl. Paul Bau
                            </p>
                        </note>
                    </seg> mit seinem Nachbarn so unzufrieden gewesen ist, aber er verlangte unbedingt, sich landsmanns
                </p>
                <p>(Zurufe: Doch.)</p>
```



Wenn wir von XML reden, reden wir von…

„Familienbeziehungen“:

- Eltern – Kinder
- Vorfahren – Nachfahren
- Geschwister

XPath Datenmodell (ab XPath 2.0)

- Elemente / Attribute / Knoten
- Alle Bestandteile eines XML-Dokuments

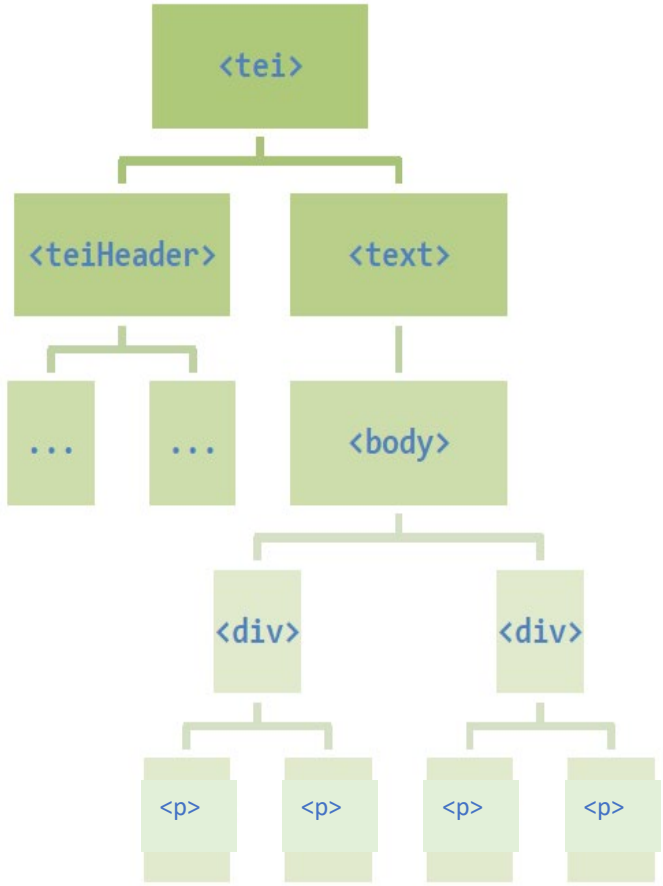
werden als Knoten (node) bezeichnet.

- Knoten (Elemente, Attribute) oder atomare Werte

(Strings, Zahlen, etc.) werden als „Items“ bezeichnet.

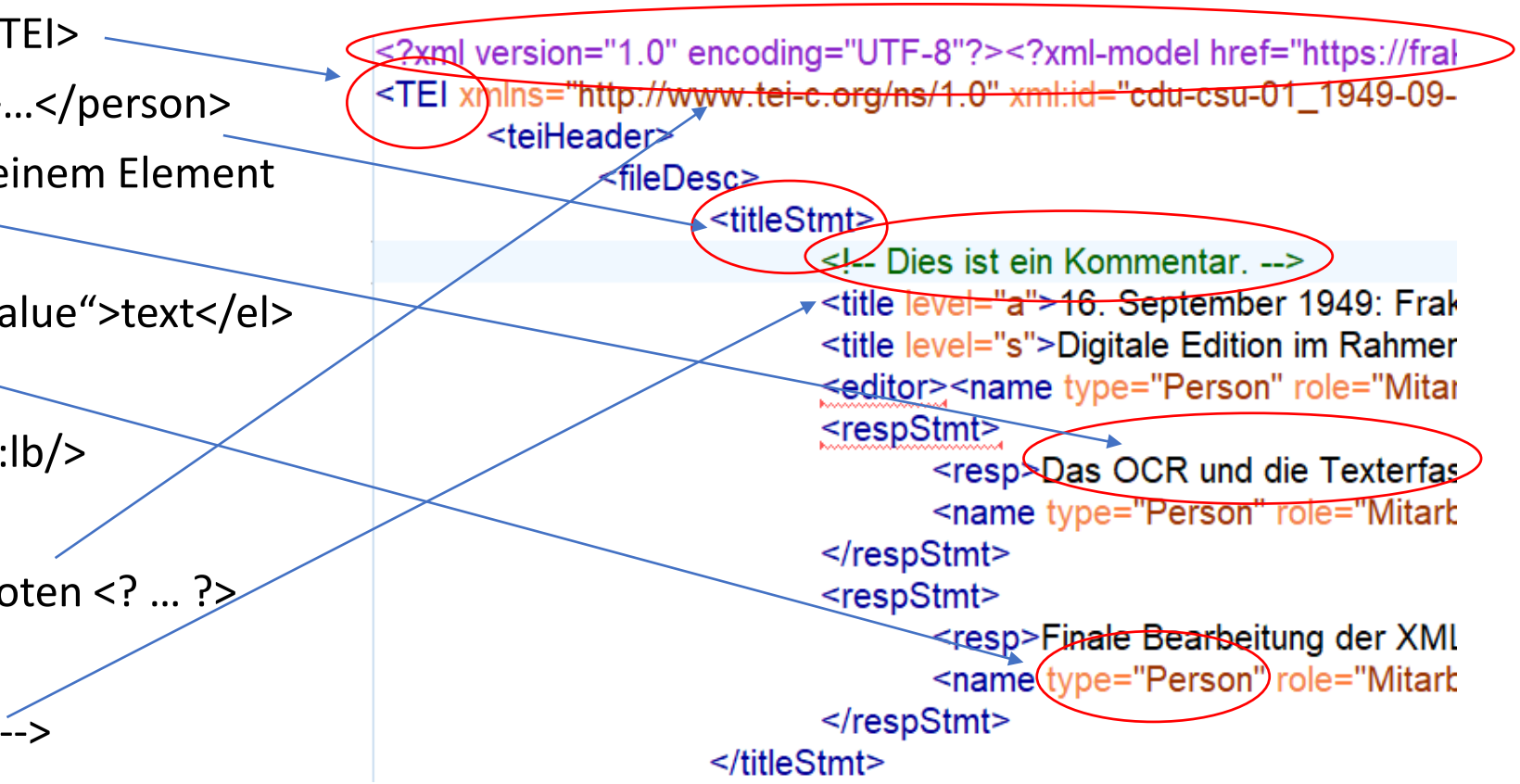
- Eine (nicht hierarchische) Reihe „Items“ wird

als **Sequenz** bezeichnet.



Knotentypen

- Dokumentknoten: das gesamte XML-Dokument („unsichtbar“)
- Wurzelknoten: <TEI> … </TEI>
- Elementknoten: <person>…</person>
- Textknoten: Textinhalt in einem Element
- Attributknoten: <el att=„value“>text</el>
- Namensraumknoten: <**tei**:lb/>
- Processing-Instruction-Knoten <? … ?>
- Kommentarknoten <!-- … -->



Bewegung im Baum…

- Lokalisierungsschritte
  - Prinzip: Achse+ Knotentest+ Prädikat
  - Syntax: achse::knotentest[Prädikat]

- Lokalisierungspfade
  - [Schritt]/[Schritt]/[Schritt]
  - /TEI/text/body/div

- „Kontext“
  - absolute Pfade (vom Dokument ausgehend) („Von ganz oben“)
  - vs. relative Pfade (vom aktuellen Kontext/Knoten ausgehend)

Auf Achse…

Vertikale Achsen

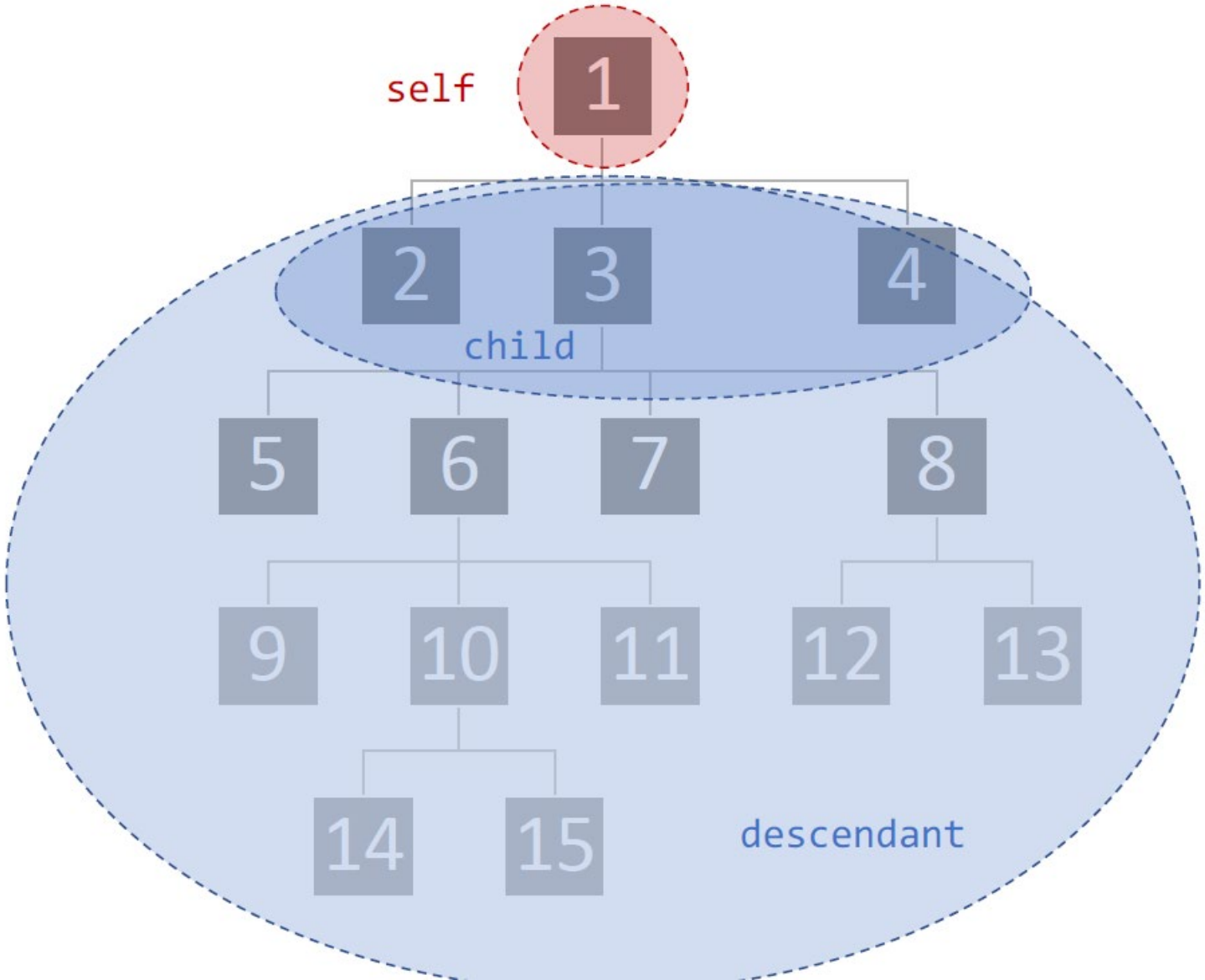
- self::
- child:: descendant:: descendant-or-self::
- parent:: ancestor:: ancestor-or-self::

Horizontale Achsen

- following:: following-or-self:: following-sibling::
- preceding:: preceding-or-self:: preceding-sibling::

Auf Achse…

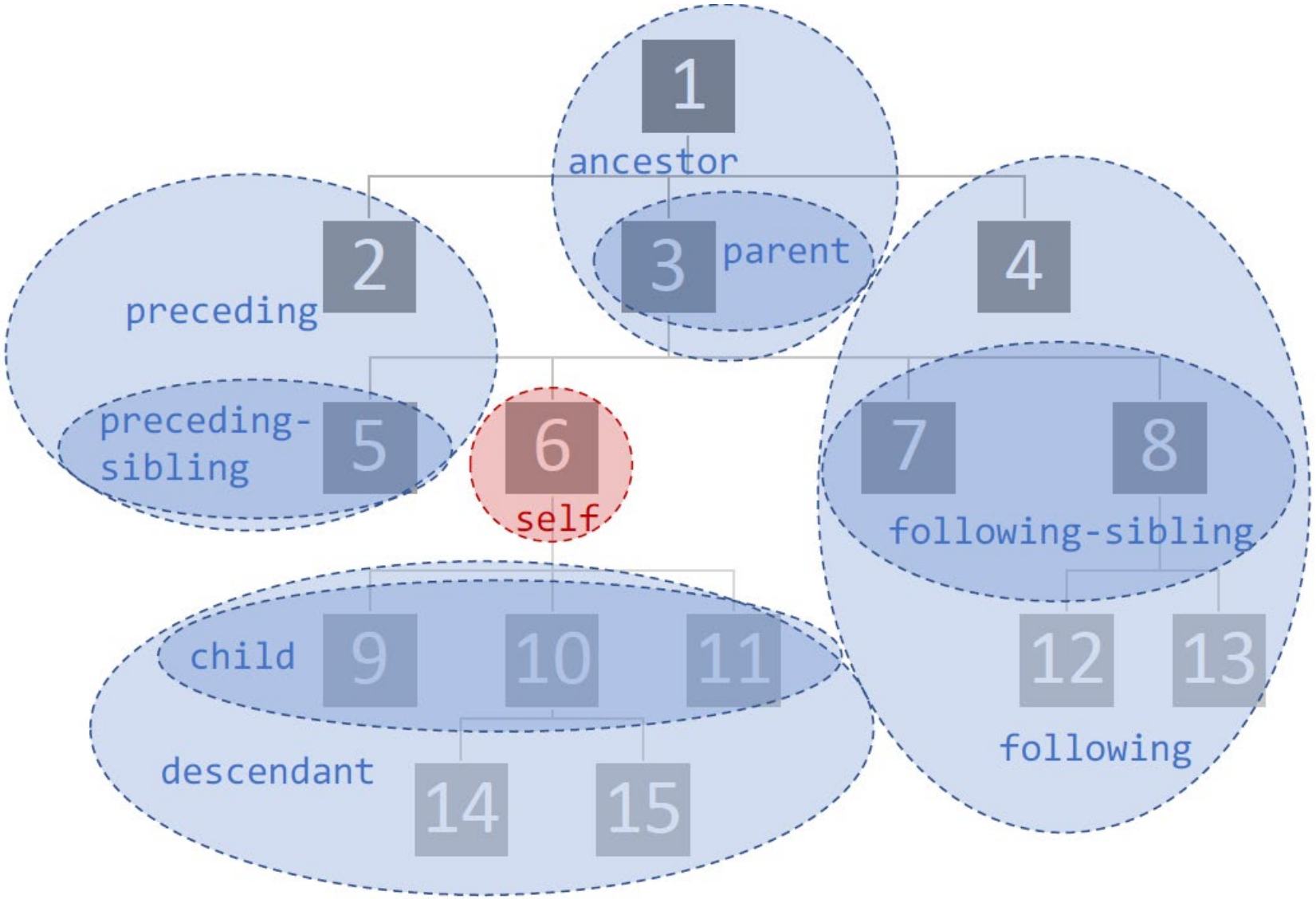
- Selbst
  - self
- Eltern / Kind
  - parent/ child
- Vorfahren / Nachfahren
  - ancestor/descendant



Auf Achse…

- Geschwister
  - preceding-sibling
  - following-sibling

- Sequenz statt Baum
  - preceding
  - following



Verkürzte Syntax

Nach dem Prinzip: **achse::knotentest([prädikat])**

- /self:: → .
- /child::persName → /persName oder persName
- /descendant::persName → //persName
- /parent::* → ..
- /child::person/attribute::role → /person/@role

- Knotentest, beliebiger Elementname → *
- /self::TEI/child::* → ./*

Syntax Zusammenfassung

- Verkettete Lokalisierungsschritte **/…/…/**
  → TEI/text/body/div//name
- Bedingungen, Prädikate **[……]**
  → //name[@type='Person’]
  → //div[3]/p[2]
- Klammern, Schachtelung **(…(…))**
  → //name[((@type='Person') and (@role='Erwaehnung’)) or (@type=Institution')]
- Operatoren **and or | = != < > + -* div**
  → //name[@ref !='#’]
- Funktionen ***Funktionsname*(*Argument*, *Argument, …*)**
  → //name[last()]/@ref
  → //name[substring-after(ref, ‘#’)

Syntax: 'Strings' in Anführungszeichen! Zahlen nicht!

Ein paar Regeln…

- Der letzte bzw. äußerste Schritt bestimmt, was ein XPath-Ausdruck zurückgibt.
- Pfade werden von links nach rechts abgearbeitet.
- Klammern werden von innen nach außen aufgelöst.

Eingabe- und Rückgabewerte

Sequenzen aus ein oder mehreren:

- Knoten (Elemente, Attribute, Textknoten...)
  - //lg à (<lg>, <lg>, <lg>, <lg>)
- Zeichenketten (Strings)
  - persName/concat(forename, “ “, surname) à („Horst Müller“)
- Zahlen
  - count(//lg) à (4)
- Wahrheitswerte (booolean: wahr / falsch)
  - contains(“Schnecke“, “ecke“) à(wahr)

Sequenzen sind **geordnet** (haben eine bestimmte Reihenfolge) und können **Duplikate** enthalten.

Sequenzen können **leer** sein (Leere Sequenz: ())

Funktionen

- Funktionen können mit ihrem Namen aufgerufen werden.
- Funktionen bestehen aus Ihrem Namen und runden Klammern: *funktion*()
- Manche Funktionen erwarten in der Klammer die Übergabe von "etwas"
  - Das kann ein Knoten sein, ein Knotensatz, ein String, eine Zahl …
- Die Übergaben sind durch Kommata getrennt
  - funktion(parameter,parameter)
- Funktionen geben dann etwas zurück
  - das kann ein Wahrheitswert sein, eine Zahl, ein String, eine Sequenz …
- 4 Gruppen von Funktionen
  - Knotenmengenfunktionen
  - Zeichenkettenfunktionen
  - Logische (boolsche) Funktionen
  - Numerische Funktionen

Funktionen - Beispiel

- contains(string,string)
  - prüft, ob ein Element oder String (das erste Argument) einen anderen String (das zweite Argument) enthält
  - liefert einen Wahrheitswert zurück
  - Auf Deutsch: Enthält die erste Zeichenkette die zweite? Wahr oder falsch? True / False?

- contains('Schnecke','ecke')
  - Deutsch: Enthält der String Schnecke den String ecke?
  - Rückgabe: true

- //name[@type='Person'][contains(., 'Schmid')]
  - Deutsch: Gibt es in meinem Baum ein Element name, das vom Typ „Person“ ist und den String Schmid enthält?
  - Rückgabe: Alle Elemente name, für die das Prädikat "true" zurückgibt (Knoten-Sequenz)

String-Funktionen I

- concat*(string, string, string,…)*
  - beliebig viele Strings miteinander verknüpfen
- substring*(string, number, number?)*
  - aus einem String einen Teil (ab einer bestimmten Position, optional bis zu einer bestimmten Position) extrahieren
- translate*(string, string, string)*
  - ersetze bestimmte Zeichen (2) in einem String (1) durch andere Zeichen (3)

String-Funktionen II

- starts-with(string,string) bzw. ends-with(string,string)
  - beginnt bzw. endet der erste String mit dem zweiten?
- substring-before(string,string) bzw. substring-after(string,string)
  - Teil des ersten Strings vor bzw. nach dem Vorkommen des zweiten Strings
- upper-case(string) bzw. lower-case(string)
  - gibt den String in Groß- bzw. Kleinbuchstaben zurück
- lower-case*(string)*
  - wandle alle Zeichen des Strings in Kleinbuchstaben um
- string-join(string*, string)
  - verknüpfe eine Menge von Strings durch ein bestimmtes Zeichen

String-Funktionen III

- matches(string,regex)
  - überprüft, ob ein String einem bestimmten Muster (regulärer Ausdruck!) entspricht
- replace(string,regex,string)
  - ersetzt Teile des ersten Strings, die dem regulären Ausdruck entsprechen, durch den zweiten String
- tokenize(string,regex)
  - teilt einen String in eine Reihe (Sequenz) von Strings, an der Stelle, die vom regulären Ausdruck bestimmt wird (das Muster gibt den „Teiler“ an)

Weitere Funktionen

- doc*(String -> URI*)
  - ein Dokument öffnen
- count(*XPath*)
  - eine Knotenmenge zählen
- distinct-values*(XPath)*
  - verschiedene Werte einer Knotenmenge ermitteln
- position()
  - die Position des aktuellen Knotens ermitteln
- data(XPath)
  - gibt die atomaren Werte aller Items in der Sequenz zurück, die der XPath-Ausdruck liefert

Ausdrücke

Muster	Beispiel
integer **to** *integer* – Range-Ausdruck: eine Sequenz von Ganzzahlen, die nacheinander durchlaufen werden	**for** $num **in** 1 **to** 10 **return** //p/name[$num] – For-Schleife in XPath; hier: Für jede Zahl von 1 bis 10 wird das Element name an der entsprechende Position zurückgegeben
if (XPath) **then** XPath **else** XPath – If-Anweisung: Ausführung von Anweisungen in Abhängigkeit davon, ob eine Bedingung erfüllt ist.	**if** (count(//p/name) gt 3) **then** „mehr als drei Namen gefunden" **else** „höchstens drei Namen gefunden" – If-Anweisung in XPath; hier: Wenn der Absatz mehr als drei Namen enthält, wird der String „mehr als drei Namen gefunden“ zurückgegeben, sonst „höchstens drei…"
for *$variablenname* **in** XPath/Sequenz **return** Sequenz – For-Schleife: gibt für jedes Item einer Sequenz etwas zurück; auf das aktuelle Item kann mit *$variablenname* Bezug genommen werden	**for** $str **in** //p/name **return** substring-before($str," ") – For-Schleife; hier: von allen Namen des Absatzes (= allen Elementen p) wird das erste Wort (= Teil-String bis zum ersten Leerzeichen) zurückgegeben

RegEx – Regular Expressions

Reguläre Ausdrücke (Regular Expressions, Regex) sind Muster, die Zeichenketten beschreiben. Sie dienen dazu, Zeichenfolgen, die diesem Muster entsprechen, zu suchen und zu bearbeiten (z. B. sie aufzuspalten oder Teile von ihnen zu ersetzen). Reguläre Ausdrücke werden u. a. in bestimmten XPath-Funktionen zur Verarbeitung von Zeichenketten verwendet.

- **Analyse** von Text,
  - z. B. *Enthält ein Textknoten eine Datumsangabe?*

    matches(text(), "\d{1,2}\.\d{1,2}\.\d{4,4}“)

- **Manipulation**
  - z.B. *normalisiere Datumsangabe 30.09.2018 zu **„2018-09-30“***

    matches(text(), "\d{1,2}\.\d{1,2}\.\d{4,4}“)

    replace(“30.09.2018“,"(\d{1,2})\.(\d{1,2})\.(\d{4,4} ","$3-$2-$1")

RegEx I

Regex	Treffer	
Kreti	**Kreti,Kreti?**	Ein regulärer Ausdruck mit den „Literalen“ K,r,e,t,i, die direkt übereinstimmen müssen. Treffer sind alle Strings, die diese Zeichenfolge enthalten
^Kret.$	**Kreti,Kreta**	Regex mit drei „reservierten Zeichen“: Zwischen Anfang (^) und Ende ($) des Strings sollen Kret und ein beliebiges Zeichen (.) stehen
Kreti\?	**Kreti?**	Ausdruck mit Literalen und einem „Quantor“ (?) , der durch die Maskierung (\) als normales ? behandelt wird
.*[0-9]+	**September 1918**	Ein beliebiges Zeichen beliebig oft (.*), gefolgt von Zahlen (beschrieben als „Bereich“ in einer in Klammern stehenden „Zeichenauswahl“:[0-9]), die ein- oder mehrmals (+) vorkommen können

RegEx II

Regex	Treffer	
\w{2}eti	**Kreti,Pleti,Kreti?**	Gesucht werden alle Strings, die aus zwei (Quantor: {2}) Zeichen der Gruppe der Wortzeichen (\w) bestehen, gefolgt von den Buchstaben eti
^[^e]+$	**Hinz und Kunz,Hinz und Hinz**	Zwischen Anfang und Ende des Strings sollen ein oder mehrere Zeichen stehen, die nicht e sind ([^e])
Kreta|Kunz	**Kreta,Hinz und Kunz**	Entweder die Zeichenfolge Kreta oder (|) Kunz
([A-Za-z]+)\s([0-9]+)	**September 1918**	Ein oder mehrere Groß- oder Kleinbuchstaben ([A-Za-z]), gefolgt von einem Leerzeichen (\s), gefolgt von einer oder mehreren Zahlen

RegEx II

- „Literale“ stehen für sich selbst und müssen direkt übereinstimmen

- Es gibt „reservierte Zeichen“ mit besonderer Funktion, z. B.

.	beliebiges Zeichen
^	Anfang eines Strings
$	Ende eines Strings
\	maskiert reservierte Zeichen

  Die reservierten Zeichen müssen mit dem Backslash „maskiert“ werden, wenn sie wörtlich gemeint sind, z. B. \? und \\

- „Quantoren“ stehen nach dem Ausdruck, auf den sie sich beziehen:

ohne	kommt genau einmal vor
?	kein- oder einmal
+	einmal oder mehrmals
*	beliebig oft
{n}	n mal
{n,}	n bis mehr als n mal
{,m}	weniger als m bis m mal
{n,m}	n bis m mal

- Eine „Zeichenauswahl“ zeigt an, dass eines von mehreren Zeichen vorkommen kann. Sie wird in eckigen Klammern notiert. Neben Literalen (z. B. [abc]) und Zeichengruppen (z. B. [\s]) kann sie auch „Bereiche“ enthalten, z. B. [a-z],[0-9]. Ein am Anfang in der Auswahl stehendes ^ bedeutet „Nicht...“

- „Zeichengruppen“ repräsentieren mehrere ‚verwandte‘ Zeichen oder ihre Umkehrung (in Großbuchstaben):

\d,\D	digit: Ziffer bzw. alles, was keine Ziffer ist
\w,\W	word: Buchstabe, Zahl oder Unterstrich
\s,\S	space: Leerzeichen, Tab, Zeilenumbruch

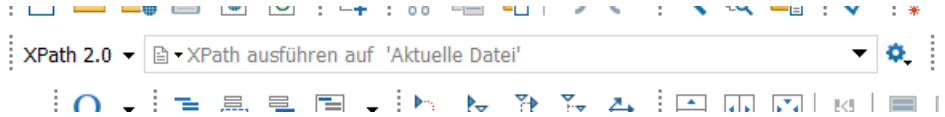
- Gruppierung von Mustern mit (...). Beim Suchen referenziert \1 die erste Gruppe, \2 die zweite, usw.; beim Ersetzen $1 die erste Gruppe, $2 die zweite, usw.

- Oder-Verknüpfungen von Mustern mit dem Pipe-Zeichen: |

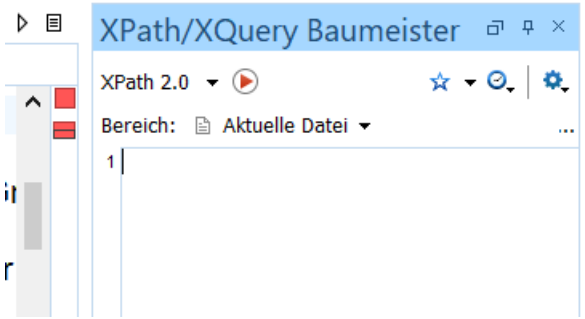
Xpath und oXygen

Es gibt mehrere Möglichkeiten, mithilfe von Xpath Dokument-Abfragen durchzuführen

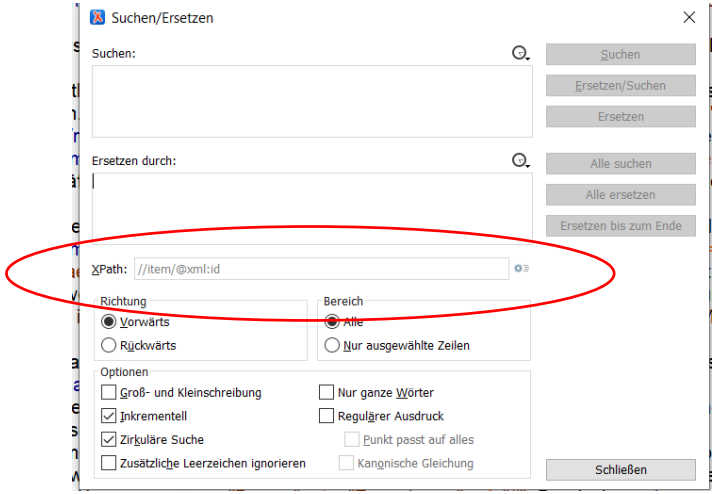
- XPath Toolbar



- Xpath/Xquery Baumeister (builder) (Fenster -> Ansicht zeigen)



- Suchen/Ersetzen Menü



Übungen I

- Übungsdatei **SchulungenKGParl/Übungsdateien/XPath_1.xml** öffnen

- Wie lautet/n die Überschrift(en) des Dokuments?

  /TEI/text/body/div/head

  -> Es gibt aber zwei <head>-Elemente, wieso wird nur eins gefunden?!

- Gib mir alle (?) Personen zurück

  /**TEI/text/body/div/p/name** oder **//name** oder **//div/p/name**

Übungen II

- Welche ist die dritte Person im Absatz 8?

  //div/p[8]/name[position()=3] *oder* **//div/p/name[3]**
- Gib mir alle segment-Elemente zurück, die eine note des Typs „comment“ haben.

  //seg[note/@type='comment']
- Gib mir alle paragraph-Elemente zurück, die solch ein segment-Element enthalten.

  //p[seg[note/@type='comment']]
- Gib mir alle Personen zurück, die mit V anfangen.

  //name[@type='Person'][starts-with(.,'V')]
- Gib mir die unterschiedlichen als Personen ausgezeichnete Strings zurück.

  distinct-values(//name[@type='Person'][starts-with(., 'B')])

Übungen III

- Gib mir alle Anmerkungen zurück, die keine editorischen Kommentare sind

  //note[*not (contains(@type,'comment'))]*
- Zu welchen name-Elementen **im Protokoll-Text** liegt keine Referenz zum Registereintrag vor?

  //body//name[@ref='#'] (vgl. **//name[@ref='#'] *)***
- Wie viele sind das?

  count(//body//name[@ref='#'])
- Zu welchen Personen im Protokolltext liegt keine Referenz vor? (Unterschied zur vorletzten Abfrage?)

  distinct-values(//body//name[@ref='#'])
- Wieviel Prozent der Liste haben keine Referenz auf das Personenregister?

  count(//body//name[@ref='#']) * 100 div count(//body//name)
- Gibt mir Personen zurück, die eine Referenz besitzten, die jedoch nicht den Nachnamen enthält (d.h. formal falsch sind)

  //name[*@type='Person'][not(contains(@ref, text())) and not(@ref='#')]*
- Gib mir alle notes mit einer nicht den Vorgaben entsprechenden ID zurück

  //body//note[*not(matches(@xml:id, "\w{3}-\w{3}-(\d){2}_(\d){4}"))]*

Übungen IV

- Welche Fragen habt ihr?

Literatur

- Übersichten
  https://www.data2type.de/xml-xslt-xslfo/xpath/referenz/
  https://www.w3schools.com/xml/xsl_functions.asp
  https://www.w3schools.com/xml/xpath_intro.asp
- Referenzen
  https://www.w3.org/TR/xpath-31/
  https://www.w3.org/TR/xpath-functions-31/
  https://www.w3.org/TR/xpath-datamodel-31/

Quellen

- P. Sahle, „XML als Datenstruktur: Einführung in Xpath“, Winter School 2020, Wuppertal, https://www.i-d-e.de/wp-content/uploads/2020/02/02_XML-Einf%C3%BChrung.pdf (abgerufen am 02.10.21)
- Schaeben, Zimmer, „Digitale Edition – Vertiefung und Nutzung X-Technologien“, DIE Autumn School 2018, https://www.i-d-e.de/wp-content/uploads/2015/11/XML-Flyer-Fortgeschrittene-II.pdf (abgerufen am 02.10.21)