Summer School “Digitale Edition”

Erschließung geisteswissenschaftlicher Quellen mit digitalen Methoden

5.-9. September 2016

Zentrum für Informationsmodellierung
Austrian Centre for Digital Humanities
Elisabethstraße 59/III, SR 81.31

Programm

Montag 5.9.

9:00-10:30	*Digitale Edition – Grundlagen und Workflow* Patrick Sahle
11:00-12:30	*Textkodierung mit XML* Markus Schnöpf und Christiane Fritze
14:30-16:00	*TEI – Einführung* Markus Schnöpf und Christiane Fritze
16:30-18:00	*Übung: XML/TEI* Markus Schnöpf und Christiane Fritze
18:30	Abendvortrag und Empfang *Historical sources through the lenses of digital humanities projects: what do we see?* Arianna Ciula

XPath

Patrick Sahle

Programm

- Wozu XPath?
- XPath als Standard
- Schachteln, Hierarchien, Bäume
- Konzepte und Grundbausteine
- Übungen

Was ist XPath

- "XPath is a language for addressing parts of an XML document” (XPath Specifications)
- XPath dient der Navigation in XML-Dokumenten und der Erzeugung von “Rückgaben”

- XPath wird vor allem in anderen X-Technologien verwandt: XSLT, XQuery

Wo findet man XPath?

- Die XPath-Spezifikation wird vom W3C (World Wide Web Consortium) gepflegt: http://www.w3.org/TR/xpath/
- Einfache Referenz: http://www.w3schools.com/xsl/xsl_functions.asp
- Einfacher Einstieg: http://www.w3schools.com/xsl/xpath_intro.asp

- Versionen: XPath 1.0, XPath 2.0, XPath 3.0

Schachteln, Hierarchien, Bäume

Sie haben es schon gelernt:

- Text ist sequentiell
- Tags + Inhalt = Elemente
- Elemente in Elementen
- Ein äußerstes Element = *Eine* Hierarchie!
- Ein „Baum“?

XML als Baum

Beispieldokument

spielcorpus-weber.xml

spielcorpus-weber.xml

```
141          </facsimile>
142          <text type="letter">
143              <body>
144                  <div type="writingSession" n="1">
145                      <opener>
146                          <salute>Lieber <persName key="A000584">Gänsbacher</persName>!</salute>
147                      </opener>
148                      <p>
149                          <ptr type="start" target="#C_01"/>Dein Brief<note xml:id="C_01"
150                              type="commentary" resp="JV">Den Eingang des nicht ermittelten Briefs
151                              vermerkt Weber im TB am 24. Februar in Aschaffenburg.</note> hat mir
152                          viele Freuden gemacht, indem ich deine schöne <ptr type="start"
153                              target="#C_02"/>Aufnahme bey <persName key="A000460"
154                              >Esterhazi</persName>
155                          <note xml:id="C_02" type="commentary" resp="JV">In den <hi rend="italic"
156                                  >Denkwürdigkeiten</hi> erwähnt Gänsbacher die Reise nach Wien Anfang
157                              1811 nur am Rande; ein Besuch im Hause des Fürsten Nikolaus Esterházy
```

Text Raster Autor

Terminologie

Elemente / Knoten

Eltern – Kinder

Vorfahren – Nachfahren

Geschwister

Wurzel, Dokument(-knoten)

Knotentypen

Dokumentknoten

Wurzelknoten

Elementknoten

Attributknoten

Textknoten

Bewegung im Baum

Lokalisierungsschritte

„gehe von hier nach da“

Knotentests

„bist Du der, den ich suche?“

Lokalisierungspfade

[Schritt]/[Schritt]/[Schritt]

/teiCorpus/teiHeader/titleStmt

„Kontext“

absolute Pfade (vom Dokument ausgehend) vs.
relative Pfade (vom aktuellen Kontext ausgehend)

Bewegung im Baum: Achsen

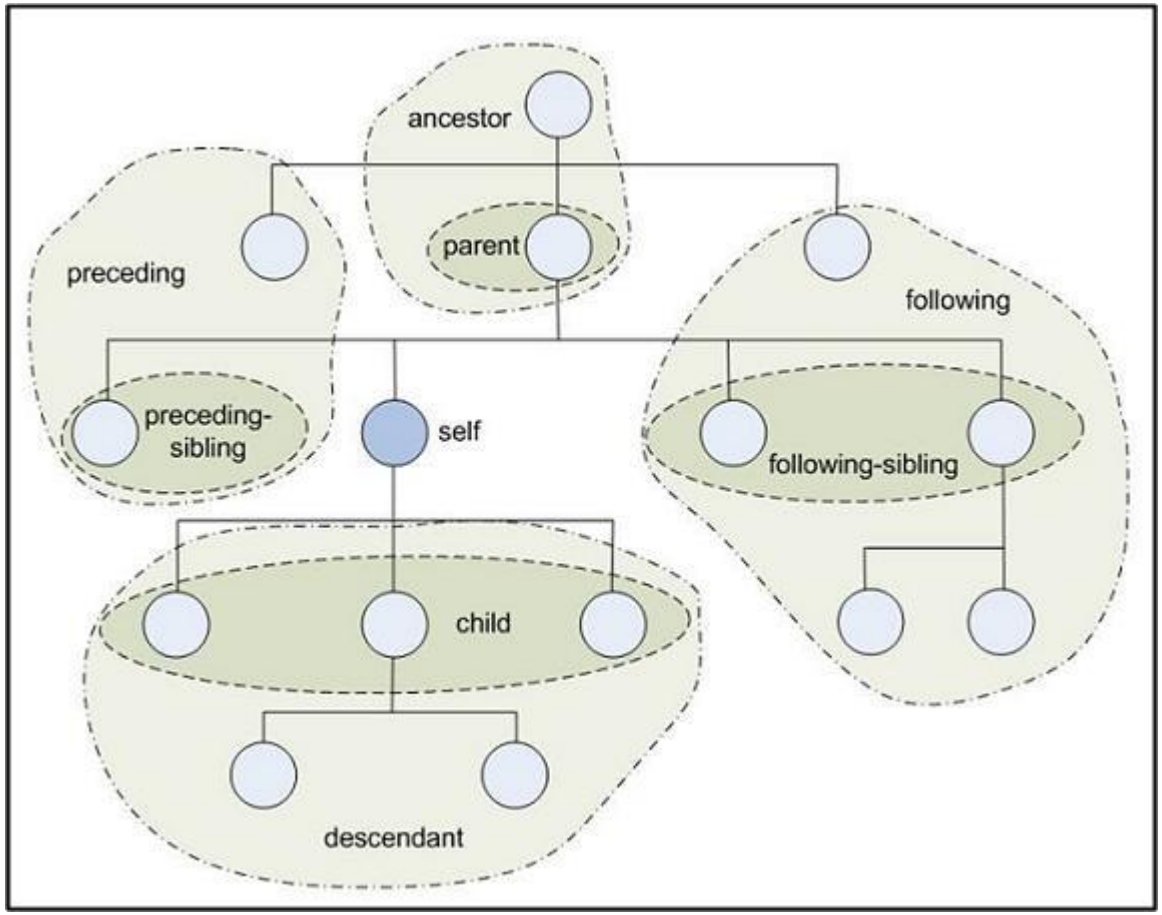
Vertikale Achsen

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

Horizontale Achsen

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

Bewegung im Baum: Achsen



Achsen: abgekürzte Syntax

self::	.
child::	/Elementname
parent::	/..
descendant-or-self::	//Elementname
attribute::	/@Attributname
Knotentest, beliebiger Elementname	*

Bewegung im Baum

Lokalisierungsschritte

Prinzip: Achse + Knotentest + Prädikat

Syntax: achse::knotentest[Prädikat]

Beispiel

```
//body/descendant::persName[@key='A000584']
```

Der Grundbaukasten

- Verkettete Lokalisierungsschritte /……/
- Bedingungen, Prädikate […….]
- Klammern, Schachtelung (…(…))
- Operatoren

  and or | = != < > + - * div
- Funktionen

  Funktionsname(*Argument*, *Argument*, …)

Rückgaben

XPath-Ausdrücke ergeben Rückgaben verschiedenen Typs:

- Elemente
- Knotenmengen
- Zahlen
- Strings
- Wahrheitswerte / Boolean
- Sequenzen

Funktionen: ein Beispiel

contains(string,string)

- prüft, ob ein Element oder String einen anderen String enthält; liefert einen Wahrheitswert zurück. Auf Deutsch: Enthält der erste String den zweiten? Ja oder nein?

- contains('Schnecke','ecke')
  → Enthält der String Schnecke den String ecke?
  → Rückgabe: true
- contains(//correspAction/date,'1810')
  → Enthält das Element date als Kindelement von correspAction (das in beliebiger Tiefe des Baumes) den String 1810?
  → Rückgabe: false
- //TEI[//correspAction[contains(date,'1811')]]
  → Rückgabe: Alle Briefe aus dem Jahr 1811 (Knotenset!)

Funktionen: ein Beispiel

contains(string,string)

- //correspAction/date[contains(.,'1811')]
  → alle Elemente „date", die den String ‚1811' enthalten
- //*[contains(date,'1811')]
  → alle Elemente, die ein Element date enthalten, das 1811 enthält
- //body/opener[contains(.,'Lieb')]/persName
  → das persName-Element in openern, die „Lieb" enthalten

Einige Funktionen I

- count(nodeset)
  - zählt etwas, erwartet eine Sequenz oder ein Knotenset, liefert eine Zahl zurück
- position()
  - gibt die Position eines Knotens an, liefert eine Zahl zurück
- string-length(string)
  - zählt die Länge eines Strings, liefert eine Zahl zurück

Einige Funktionen II

- starts-with(string, string)
  - prüft, ob ein String mit einem anderen String beginnt, liefert einen Wahrheitswert zurück
- not(boolean)
  - dreht einen Wahrheitswert um, liefert einen Wahrheitswert
- max(sequence of numbers)
  - ermittelt den maximalen Wert aus einer Reihe von Werte, liefert eine Zahl zurück
- distinct-values(sequence of strings)
  - gibt eine Sequenz von (unterschiedlichen) Werten zurück

Einige Funktionen III

- substring(string,start,length)
- matches(string, pattern)
- tokenize(string, pattern)
- name()
- number(string), string(number)
- doc(URI)

Übungen I

- Alle Orte in den Brieftexten

  → //body//p//settlement
- Alle Absendeorte der Briefe

  → //correspAction[@type='sent']/placeName
- Welche Ortsangaben starten mit A?

  → //settlement[starts-with(.,'A')]
- In welchen Briefen (Nummer!) hat Bearbeiter „SS“ Änderungen vorgenommen?

  → //change[@who='SS']/ancestor::TEI//publicationStmt/idno/@n
- Wieviele Briefe sind im Corpus?

  → count(//TEI)

Übungen II

- Wieviele Änderungen sind im Durchschnitt pro Brief festgehalten?

  → count(//revisionDesc/change) div count(//TEI)
- Vor wievielen Jahren wurde der erste Brief geschrieben?

  → 2016-number(//TEI[1]//correspAction [@type='sent']/substring(date/@when,1,4))
- Alle (verschiedenen) vorkommenden Personen?

  → distinct-values(//persName/@key)
  → //persName[not(preceding::persName/@key) = @key]//text()
- Was ist die Hierarchietiefe in den Dokumenten?

  → max(//*/count(ancestor::*))