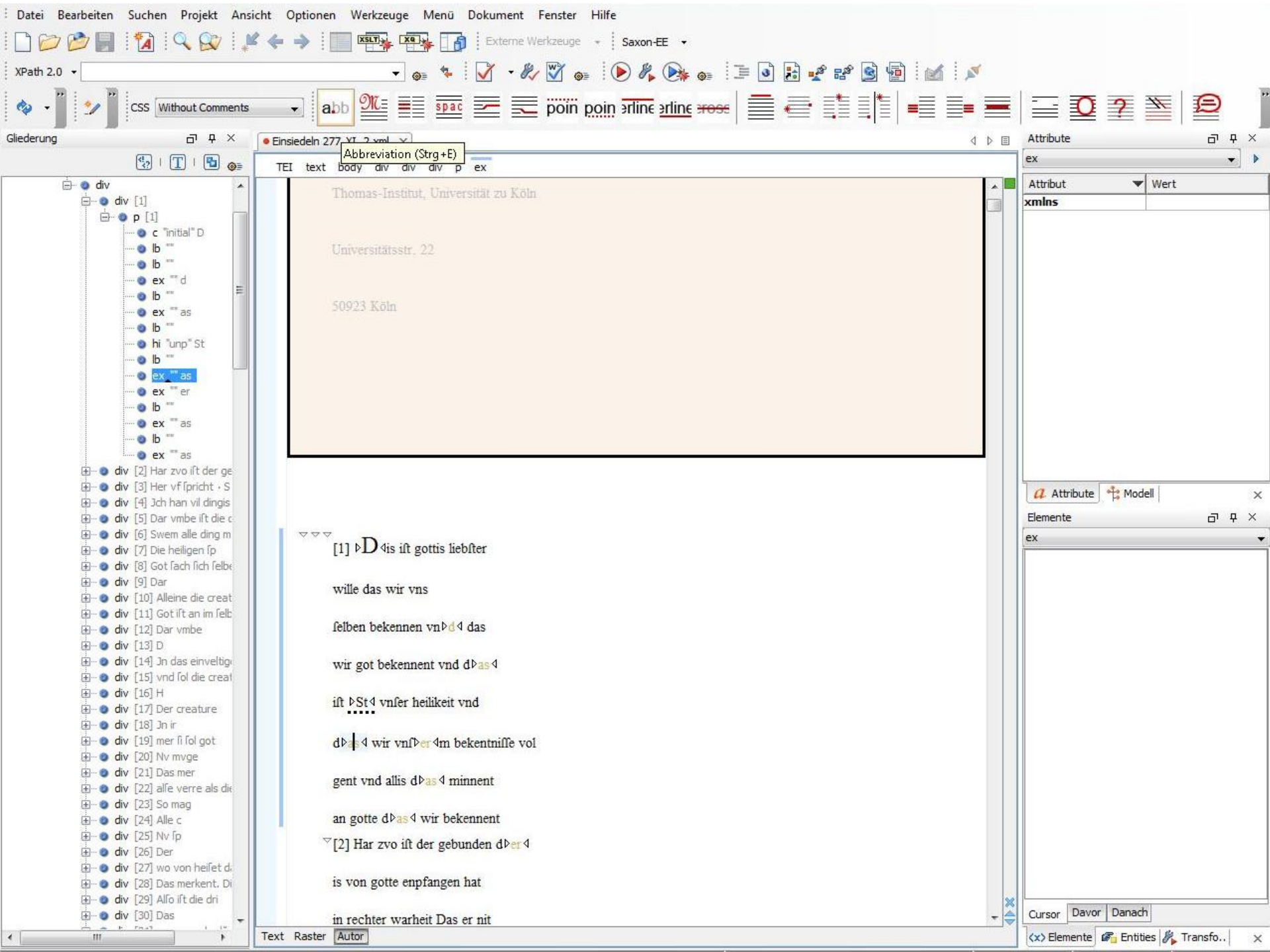
XML – XPath – XSLT

Wiederholung und Auffrischung

Ulrike Henny und Patrick Sahle

XML, Wiederholung

- Werkzeuge? oXygen


Datei Bearbeiten Suchen Projekt Ansicht Optionen Werkzeuge Menü Dokument Fenster Hilfe
Externe Werkzeuge
Saxon-EE
XPath 2.0
CSS Without Comments
Gliederung
Einsiedeln 277, XI, 2.xml
Abbreviation (Strg+E)
TEI text body div div div p ex
Thomas-Institut, Universität zu Köln
Universitätsstr. 22
50923 Köln
[1] Dis iſt gottis liebſter
wille das wir vns
ſelben bekennen vnd das
wir got bekennent vnd das
iſt St vnſer heilikeit vnd
das wir vnſerm bekentniſſe vol
gent vnd allis das minnent
an gotte das wir bekennent
[2] Har zvo iſt der gebunden der
is von gotte enpfangen hat
in rechter warheit Das er nit
Attribute
ex
Attribut
Wert
xmlns
Attribute
Modell
Elemente
ex
Cursor
Davor
Danach
Elemente
Entities
Transfo..
Text
Raster
Autor

XML-Wiederholung

- XML
  - Wissen codieren: Grundprinzipien, Baumstruktur, syntaktische Regeln, Wohlgeformtheit vs. Gültigkeit
- Schemasprachen
  - Daten kontrollieren: Elemente und Attribute, Inhaltsmodelle, Verschiedene Sprachen, unterschiedliche Ausdrucksmöglichkeiten
- XPath
  - Bäume navigieren: Achsen, Lokalisierungsschritte, Bedingungen, Funktionen, Operatoren
- XSLT
  - Daten transformieren: Definierte Elemente, Schablonen

XPath - Beispiele

- Pfadausdruck:
  - //kapitel//p/following-sibling::p/attribute::lang
  - .//kapitel//p/following-sibling::p/@lang
  - Achsen? Abgekürzte Schreibweisen? Flyer!
- Bedingung:
  - //titel[. = "Die Leiden des jungen Werthers"]
  - //buch[titel = “Die Leiden des jungen Werthers“]/p
- Funktionen:
  - //wort[lower-case(substring(., 1, 1)) = "ä")]
- Operatoren:
  - //wort[contains(., "ä")] and contains(., "ö") or contains(., "ü")]
  - //wort[substring(., 1, 1) != “A“]

XPath - 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

XPath - Funktionen

- String-Funktionen:
  - concat*(string, string, string,…)*
    - beliebig viele Strings miteinander verknüpfen
  - contains*(string, string)*
    - enthält der erste String den zweiten?
  - 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)

XPath - Funktionen

- String-Funktionen:
  - upper-case*(string)*
    - wandle alle Zeichen des Strings in Großbuchstaben um
  - lower-case*(string)*
    - wandle alle Zeichen des Strings in Kleinbuchstaben um
  - string-join(string*, string)
    - verknüpfe eine Menge von Strings durch ein bestimmtes Zeichen

XSLT - Ausdrücke

- <xsl:template match="*XPath-Ausdruck*">
  ... <xsl:apply-templates select="*XPath*" />
  </xsl:template>
  - Schablone für (mit @match bestimmte) Knoten. *Apply templates* sagt, dass noch andere Schablonen berücksichtigt werden sollen

- <xsl:for-each select="*XPath*">
  <xsl:sort /> …
  </xsl:for-each>
  - *For each*-Schleife: für jeden der mit @select ausgewählten Knoten wird etwas getan. Die Knoten werden (hier aufsteigend alphabetisch) sortiert.

XSLT - Ausdrücke

- <xsl:value-of select=". " />
  - der Wert des ausgewählten Knotens/der Knoten wird ausgegeben

- <xsl:variable name= "*String*">…</xsl:variable>
  … <xsl:value-of select= "$String" />
  - Anlegen einer Variablen mit einem bestimmten Namen. Ansprechen /Ausgabe der Variablen mit vorangestelltem $-Zeichen.

XSLT

- <xsl:if test="*XPath-Ausdruck*">
  ... weitere Anweisungen
  </xsl:if>
  - die in @test angegebene Bedingung wird überprüft; wenn sie zutrifft, werden weitere Anweisungen ausgeführt

- <xsl:choose>
  <xsl:when test="*XPath-Ausdruck*"/> *[ggf. mehrfach]*
  <xsl:otherwise/>
  </xsl:choose>
  - hier werden ebenfalls ggf. mehrere Bedingungen nacheinander überprüft; wenn keine zutrifft, werden in *otherwise* angegebene Standardanweisungen ausgeführt

XSLT

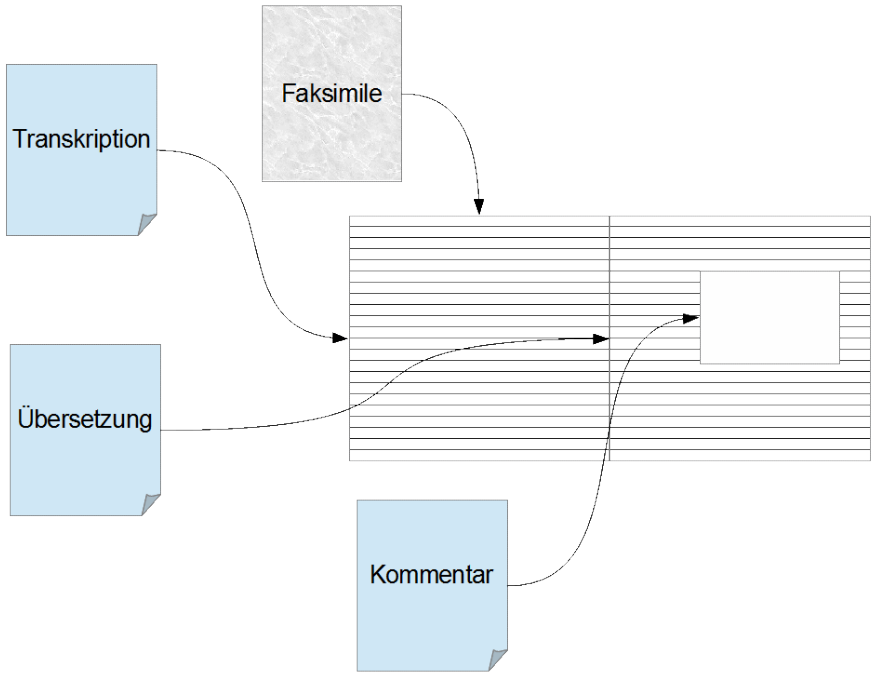
- <xsl:result-document ref=“string”/>
  - schreibe ein Ergebnisdokument in dem in @ref angegebenen Verzeichnis (Pfad), mit dem in @ref angegebenen Dateinamen
- <xsl:copy select= „*XPath*">…</xsl:copy>
  - erzeugt eine Kopie des aktuellen Knotens (ohne Kindelemente und ohne Attribute)
- <xsl:copy-of select= „*XPath*">…</xsl:copy>
  - erzeugt eine Kopie des aktuellen Knotens, seiner Attribute und aller Kindknoten (also des kompletten Teilbaums vom aktuellen Kontext aus)

XSLT

- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  - xmlns:[prefix]="*String*"
  - version="*1.0* | *2.0*"
  - xpath-default-namespace="*String*"
  - exclude-result-prefixes="*String*">
    - = Wurzelement eines XSLT-Stylesheets; @xmlns zur Angabe von Namensräumen; @version: XSLT-Version; @xpath-default-namespace: Standardnamensraum für Pfadausdrücke, mit denen im Ausgangsdokument gesucht wird; @exclude-result-prefixes: diese Namensraumpräfixe nicht ins Ergebnisdokument schreiben
- <xsl:output
  - method=„xml | xhtml | html | text">
    - legt fest, welchen Inhaltstyp das Ergebnisdokument haben soll: XML, XHTML, HTML, Text

Übung 1: Von XML in‘s Web

- Gegeben: Transkription, Übersetzung, Kommentare, Links zu Faksimiles
- Gesucht: Synoptische Webdarstellung



Übung 1: Von XML in‘s Web

- Beispiel für: Amoenitates Exoticae - Faszikel I, Relatio I
- Ausgangsdateien:
  - tei-transcript.xml
  - tei-translation.xml
  - tei-comment.xml

- Allgemeine Überlegungen: Superstruktur „digitale Edition in XML“
  - eine Datei – viele Dateien?
  - eindeutige *Identifier*!
  - Entities?
  - METS?

Übung 1: Von XML in‘s Web

- Schritte:
  - XSLT-Datei erstellen
  - Ausgangsdateien zusammenführen
  - Struktur für synoptische Darstellung schaffen:
    - HTML-Datei aufbauen
    - Slots für Transkription und Übersetzung
  - Textausgabe:
    - Texte in den Slots ausgeben
    - Templates für Inline-Markup erstellen *(Push*-Prinzip*)*
  - Verknüpfen:
    - Faksimiles verlinken
    - Kommentare anzeigen

Wie könnte das Ergebnis aussehen?

[3] Relatio I. Persona Regis Persarum, Status hodierni Regni, et regiminis ratio in genere.

§ I.

Regis Persarum magnitudo. ***Praefatio.***

Augustam Persicae Maiestatis Aulam, multitudine Purpuratorum, auro, divitiis et innumeris Regalis magnificentiae generibus fulgentem, ex vivo praesentique vultu descripturus, magni ausus videor provinciam aggredi. Si enim consignandis rebus etiam haud magni momenti, scriptorem requirimus congeneris fori, ne historiae existimatio detrimentum ex peregrino genio orationis capiat, vix ego absolvendus temeritatis fuero, qui extra oleas facultatis meae, opus tam insolentis argumenti suscipio. Cuius equidem gravitate se deterreri tenuitas mea pateretur, si fas esset ex styli verecundia praesentissimam dimittere occasionem, quae Monarchici huius theatri scenas recessusque omnes, et quicquid in iis memorabile ac a vulgi notitia remotum geritur, gratioso indultu perlustranda atque evolvenda exhibet. Accipe igitur Aulam, B. L., non qua par est facundia, sed quanta decet fide, per membra articulosque intime exponendam: Aulam inquam nostri aevi; sed, ut ab omni seculorum memoria, sive sub Monarchia Orbis, sive sub temporum sequiorum fatis illustrem, ita non minus sub recentiori Sophorum regimine felicem, floridam, pacificam, invictae hactenus potentiae, tremendam Orienti, ac toto orbe incly [4] tam. Varietatem occurrentis materiae exhaurire placet brevi serie Relationum: quia haec methodus Viatorum commentariis propria est, et legentibus delectabilior.

Potestas absolutissima.

Caput Imperii Persici *Rex* est, qui ad Dignitatis fastigium iure haereditatis provehitur. Magnitudinem Eius, cum vasta moles territorii, tum vero

[3] Bericht I. Die Person des Königs der Perser, der Zustand des heutigen Königreichs und das System der Regierungsform im Allgemeinen.

§ I.

Die Größe des Königs der Perser. ***Vorwort.***

Da ich den erhabenen Hof der Persischen Majestät im Glanze einer Menge von Purpurträgern sowie von Gold, Reichtümern und unzähligen Arten königlicher Prachtentfaltung nach dem lebendigen und gegenwärtigen Erscheinungsbild beschreiben will[K] , ist es offensichtlich, dass ich an die Aufgabe eines großen Unterneh
schon zur schriftlichen Beglaubi
Bedeutung haben, einen aus de
fordern, damit nicht die Einschä
fremdartigen Geistes der Darstellungsweise Schaden nimmt, dann werde ich selber kaum vom Vorwurf der Verwegenheit freizusprechen sein, da ich außerhalb der Ölbäume meiner Fähigkeit[K] die Bearbeitung eines so ungewöhnlichen Themenkreises übernehme. Durch dessen Bedeutsamkeit würde sich meine Wenigkeit[K] allerdings abschrecken lassen, wenn es schicklich wäre, aus Scheu vor der sprachlich-stilistischen Gestaltung eine sich unmittelbar bietende höchst günstige Gelegenheit verstreichen zu lassen, welche die Schauplätze dieser monarchischen Bühne und alle ihre Hintergründe sowie sämtliche Vorgänge aufzeigt, die dabei erwähnenswert sind und entfernt von der Kenntnis der Allgemeinheit geschehen - Ereignisse also, die daher mit wohlgefälliger Einwilligung genau untersucht werden müssen und dargelegt werden müssen. Empfange folglich, geneigter Leser, einen Königshof in einer intimen, nach einzelnen Gliedern und Teilen angeordneten Darstellung; zwar ohne die angemessene

descripturus Zur gesamten - besonders stark von Motiven der klassischen antiken Literatur geprägten - Argumentation Kaempfers in seiner Einleitung vgl. Neuhausen 2004, S. 62-69.

Baukasten

- xsl:stylesheet, @xmlns, xsl:output
- xsl:variable, document(), xsl:copy-of
- xsl:template
- xsl:apply-templates @select
- xsl:value-of, {}
- current()
- HTML, CSS, ein bisschen Javascript

- → Ergebnis (mit Erläuterungen): siehe Spieldaten.zip/Kaempfer-Beispiel/synopse.xsl

Übung 2: Register

- Prototypischer Register-Workflow?

- Transkription
- Tiefenerschließung
- Modellierung? TEI-Arsenal?
- Zentrale Registerdatei?
- Entities / festes Vokabular als Attributwert im Schema?
- Gegenseitige Kontrolle: Transkription vs. Liste
- Datenanreicherung
- Registergenerierung

Übung 2: Register Witzel

- „ich würde gern wissen, wie ich aus meinen TEI-Dateien einen Datei-übergreifenden Ortsindex erstellen kann. Im Index sollen nicht die Ortsnamen aus der Quelle erscheinen, sondern die in listPlace abgelegten normierten Ortsnamen. “

i|d|e

Wie könnte das Ergebnis aussehen?

G

Genf: Beulwitz S.42 (Genev)

Geschwenda: Andreä S.21 (Geschwenda im Arnstädtischen)

Gotha: Andreä S.21

Griesheim: Andreä S.21 (Grießheim an der Ilm)

H

Haarlem: Beulwitz S.44 (Harlem)

Hammersfeld: Andreä S.26

Helmstedt: Andreä S.23 (Helmstädt)

Herschdorf: Andreä S.20 (Herschdorff) S.21 (Herrschdorff)

Hirschberg: Beulwitz S.38 (Hertzberg)

I

Ichtershausen: Andreä S.20, S.23

J

Jena: Andreä S.23, S.26; Beulwitz S.40, S.42 S.41 (JENA)

Übung 2: Register Witzel

- „ich würde gern wissen, wie ich aus meinen TEI-Dateien einen Datei-übergreifenden Ortsindex erstellen kann. Im Index sollen nicht die Ortsnamen aus der Quelle erscheinen, sondern die in listPlace abgelegten normierten Ortsnamen. “
- Gegeben: einzelne Transkriptionen mit <placeName> und <listPlace> im Header
- Was spricht für, was gegen dieses Vorgehen?
- Strategien?
  - Strategie A: zentralisierte <listPlace>
  - Strategie B: mehrere <listPlace>

Baukasten

- xsl:stylesheet, @xmlns, xsl:output
- xsl:template, xsl:value-of,
- xsl:variable, fn replace()
- xsl:for-each, fn collection()
- xsl:for-each-group @select @group-by
  - Bezugnahme: current-group(), current-grouping-key()
- xsl:if, position(), last()
- xsl:text

Vorgehen und Pseudocode

- ...

- → Ergebnis (kommentiert): Spieldaten.zip/Witzel/register.xsl

… ist Ihnen was aufgefallen? zwei Basisstrategien in XSLT

- "pull-Ansatz"
  - ein template erledigt so viel wie möglich, indem es nacheinander Teilaufgaben abarbeitet und sich dazu die benötigten Inhalte selbst holt
  - zunächst übersichtlich, einfach zu entwickeln, unabhängig von der Reihenfolge der Elemente im xml, nicht modular, wird bei komplexen Dokumenten unübersichtlich
- "push-Ansatz"
  - templates betreffen möglichst immer nur bestimmte Elemente und verweisen für deren Inhalte auf andere templates (xsl:apply-templates)
  - für Anfänger unübersichtlicher, abhängig von der Reihenfolge der Elemente im xml, modularsierte templates werden in verschiedenen Kontexten verwendet, bei komplexen Dokumenten letztlich leichter wartbar
- in der Praxis häufig Mischung beider Strategien

XML – XPath – XSLT

Erweiterung und Vertiefung

Ulrike Henny und Patrick Sahle

Themen

- XPath
  - Ranges
  - If-Anweisungen
  - For-Schleifen
- XSLT
  - Templates
  - Gruppieren
  - Sortieren
- Regex
- Beispiele:
  - Datenanreicherung
  - Visualisierung

XPath - Ranges

- Schleife über eine Sequenz von Ganzzahlen
- Syntax:
  - *Integer* **to** *integer*
- Beispiele:
  - //absatz[position() = 1 to 3]
  - //absatz[position() = 1 or position() = 2 or position() = 3]
  - //absatz[position() = (1, 2, 3)]
  - //absatz[position() = (1,4,8)]
  - <xsl:for-each select="1 to 10">
    …
    </xsl:for-each>

XPath – If-Anweisungen

- Eine bedingte Anweisung
- Syntax:
  - **If** *(expression)* **then** *expression* **else** *expression*
- Beispiel:
  - If (count(//kapitel/absatz) gt 20)

    then "Dies ist ein langes Kapitel"

    else "Dieses Kapitel ist eher kurz"
  - <xsl:variable name="autoren" select="count(distinct-values(//autor)">

    <xsl:value-of select="if ($autoren = 0) then ‘Keine Autoren‘ else if ($autoren = 1) then ‘ein Autor‘ else concat($autoren, ‘ Autoren‘)" />

XPath – For-Schleifen

- Durchlaufen eine Sequenz; Rückgabe für jedes Element
- Syntax:
  - **for** *$variable* **in** (*item**) **return** *item**
- Beispiel:
  - for $ort in document(‘ortsregister.xml‘)
    return {$ort]
  - for $ort in document(‘ortsregister.xml‘)
    return
      if ($ort/einwohnerzahl gt 100,000)
      then {$ort/name}
      else {$ort/name}

XSLT - Stylesheets

- Einbindung anderer Stylesheets in das aktuelle Stylesheet
- xsl:include:
  - eingebundene Templates haben die **gleiche Priorität** wie die Templates im aktuellen Stylesheet
- xsl:import:
  - Eingebundene Templates haben eine **niedrigere Priorität** als die Templates im aktuellen Stylesheet
- Syntax:
  - <xsl:include href="*URI*" />
  - <xsl:import href="*URI*" />
- Top level - Elemente:
  - xsl:include und xsl:import stehen direkt unter dem Wurzelelement xsl:stylesheet

Regular Expressions (in Oxygen)

- Ausgangslage
  - Verschiedene Dialekte
  - Maßgeblich: perlre
  - Umfang der Implementierung?
  - oXygen vs. XML-Schema vs. XSLT
- Syntax

Regular Expressions (in Oxygen)

- Syntax (nur das allerwichtigste)

.	Beliebiges Zeichen	
[]	Zeichenklasse	[0-9], [a-z] beliebige Ziffer bzw. Buchstabe
()	Gruppierung	… kann später angesprochen werden
|	oder	[r | v] der Buchstabe r oder v
\	Maskierung	\. tatsächlich ein Punkt und nicht beliebieges Zeichen
*	Beliebige Häufigkeit	.* beliebige Menge beliebiger Zeichen
+	Ein oder mehrmals	[0-9]+ eine Zahl mit mindestens einer Stelle
?	Ein oder kein Mal	[0-9]? eine Ziffer oder nichts

Regular Expressions (in Oxygen)

- Syntax (schon nicht mehr so wichtig)
  - \d Ziffer
  - \w Buchstabe oder Ziffer
  - \s Whitespace-Zeichen
  - ^ Anfang einer neuen Zeile
  - $ Zeilenende
  - \n Zeilenumbruch
  - ? Greediness ausschalten ([0-9].*\])*? [in Oxygen10 nicht implementiert]

- … und weitere, siehe z.B. http://perldoc.perl.org/perlre.html

Regular Expressions (in Oxygen)

- Ausgangslage (perlre)
- Syntax
- Das wichtigste: Gruppierte Muster bilden Variablen auf die man beim Ersetzen zugreifen kann
- Obacht: Zeilenorientierung, greediness
- Beispiel (einfache Seitenumbrüche)

Regular Expressions (in Oxygen)

- Beispiel
- text <pb n="1ra" img="../images/img-1-r-a.jpg"/> text <pb n="1rb"/> text <pb n="1va"/> text <pb n="1vb"/> text <pb n="2ra"/> text
- Ziel: ergänze analog img-Attribute
- text <pb n="1ra" img="../images/img-1-r-a.jpg"/> text <pb n="1rb" img="../images/1-r-b.jpg"/> text <pb n="1va" img="../images/1-v-a.jpg"/> text
- Lösung:
  - Suche: (([\d]+)([rv])([ab]))"/
  - Ersetze durch: $1" img="../images/$2-$3-$4.jpg"/

Regular Expressions (in Oxygen)

- Ausgangslage (perlre)
- Syntax
- Das wichtigste: Gruppierte Muster bilden Variablen auf die man beim Ersetzen zugreifen kann
- Obacht: Zeilenorientierung, greediness
- Beispiel (einfache Seitenumbrüche)
- Erweiterung: regex plus XPath
- Möglichkeiten und Grenzen

regex in XSLT

- xsl:analyze-string
  - xsl:matching-substring, xsl:non-matching-substring
- XPath-Funktionen
  - matches()
  - replace()
  - tokenize()

Datenanreicherung

- Ziel: Gegebene Daten verbessern
- Methoden:
  - Systematische Veränderungen & Umbauten
  - Abgleich mit externen Daten; Nutzung von Services
    - Bedingung: Die externen Daten / Antworten müssen XML sein
  - Ggf. manuelle Nachbearbeitung

Externe Datenquellen?

- Geonames → Ids, Koordinaten
- Google → Koordinaten
- Deutsche Nationalbibliothek → GND (Personen, Schlagwörter)
- Wikipedia
  - Seiten → Infoboxen, Kategorien, freie Textinhalte
  - Tools → z.B. PND-Nummern

Übung 1: Datenanreicherung

- Beispiel: Ortsregister
  - Geokoordinaten sammeln mit Google-Maps…
- XSLT:
  - copy all but ...
  - Templates für die Elemente, die verändert werden
  - Web-Service-URL, Namensräume?
  - document()
  - ggf. Stylesheet rekursiv aufrufen

→ Ergebnis (kommentiert): spieldaten.zip/Witzel/geokoordinaten.xsl

Copy all, but ...

- Grundtemplate

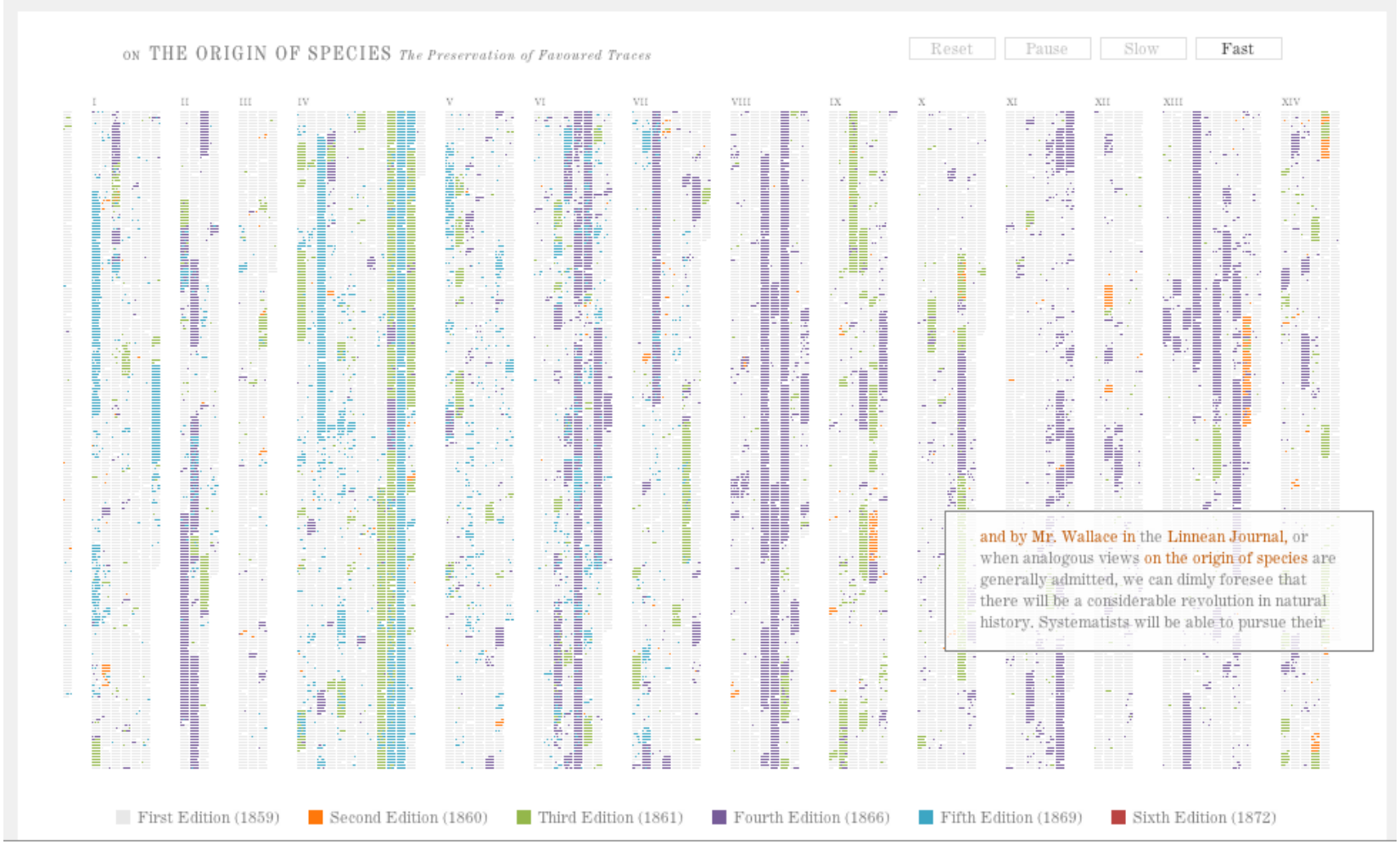
```
<xsl:template match="node() | @* | processing-instruction() | comment()">
    <xsl:copy>
        <xsl:apply-templates select="node() | @* | processing-instruction()
          | comment()"/>
    </xsl:copy>
</xsl:template>
```

Was ist der Trick dahinter?

Übung 2: Visualisierung

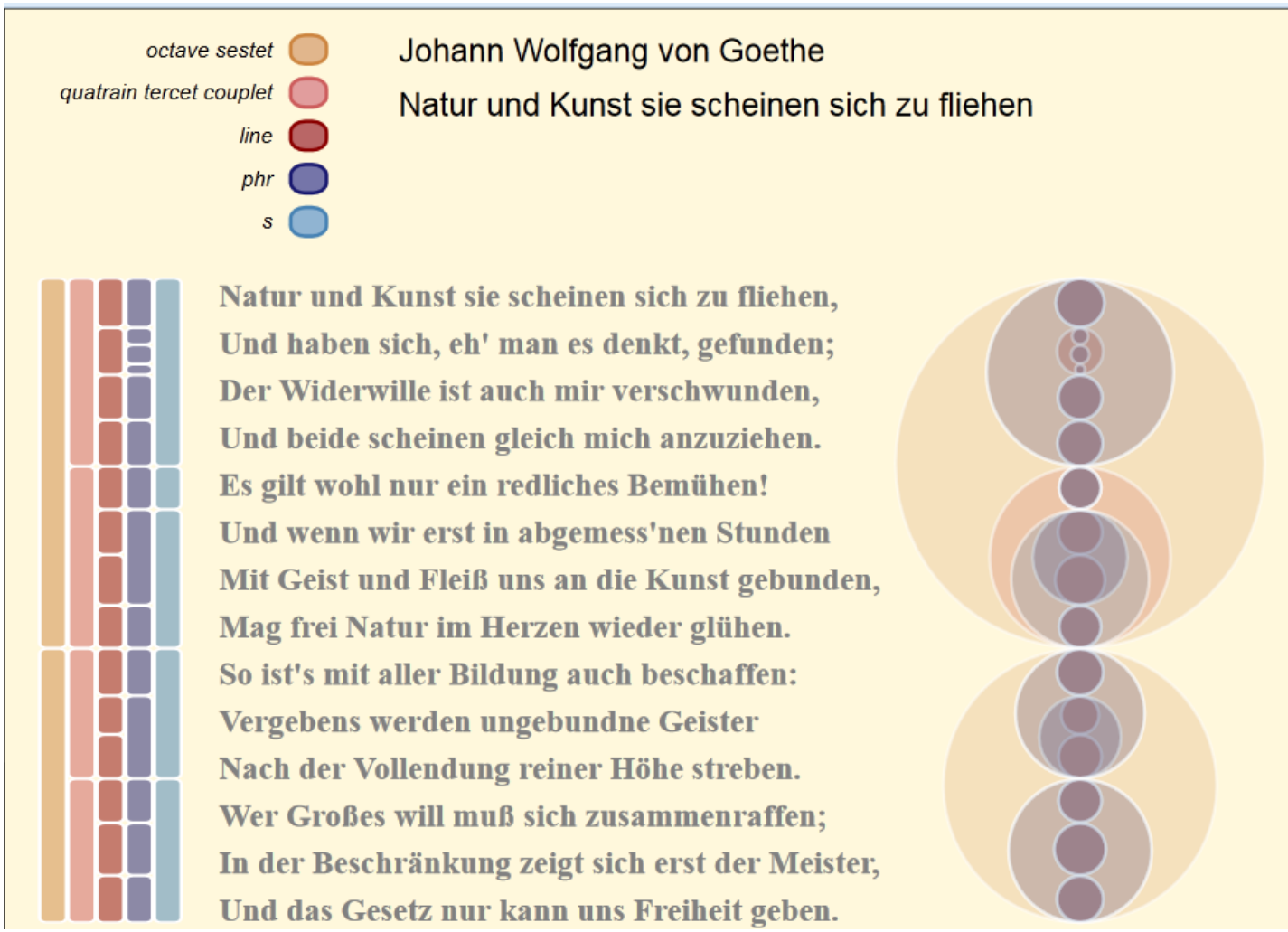
- Ziele:
  - visuellen Zugang schaffen
  - Browsing erleichtern / Strukturen zeigen
  - neue Erkenntnisse/Fragen?
- Methoden:
  - geeignete Visualisierung?
  - Visualisierungswerkzeug finden
  - Daten aufbereiten

Visualisierung?



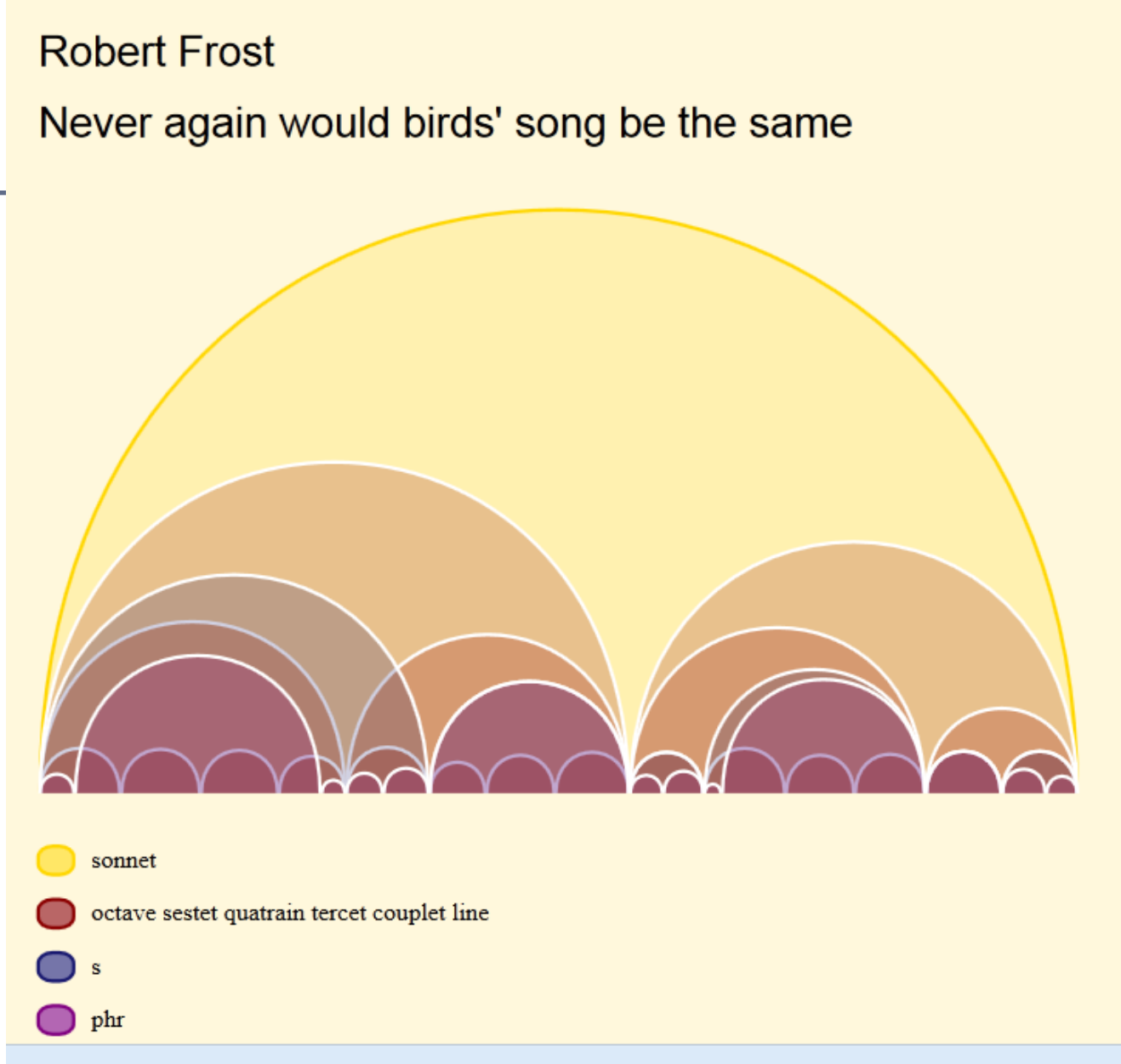
http://benfry.com/traces/

Textstrukturen

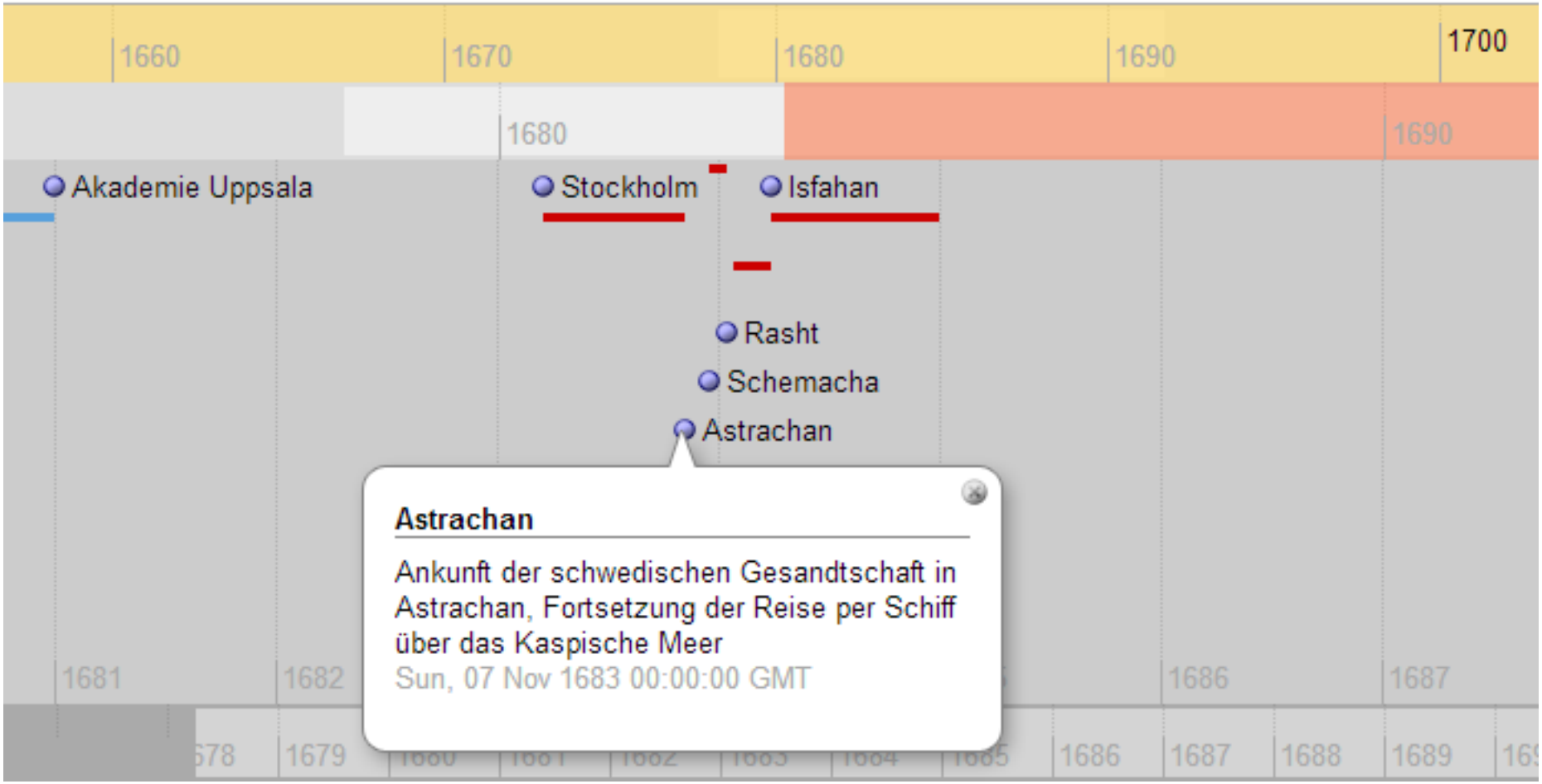


http://www.piez.org/wendell/papers/dh2010/clix-sonnets/index.html

Textstrukturen

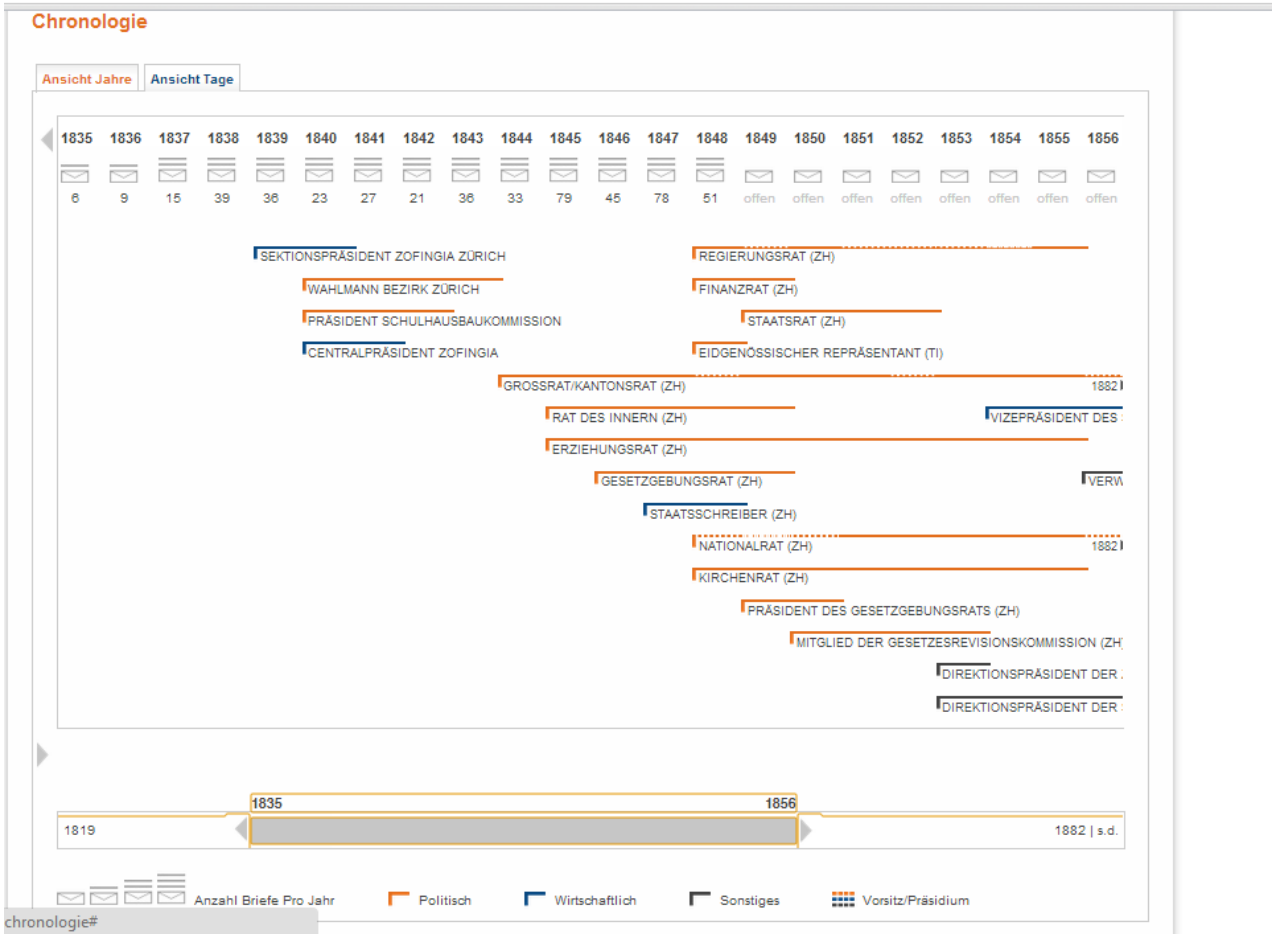


http://www.piez.org/wendell/papers/dh2010/clix-sonnets/index.html

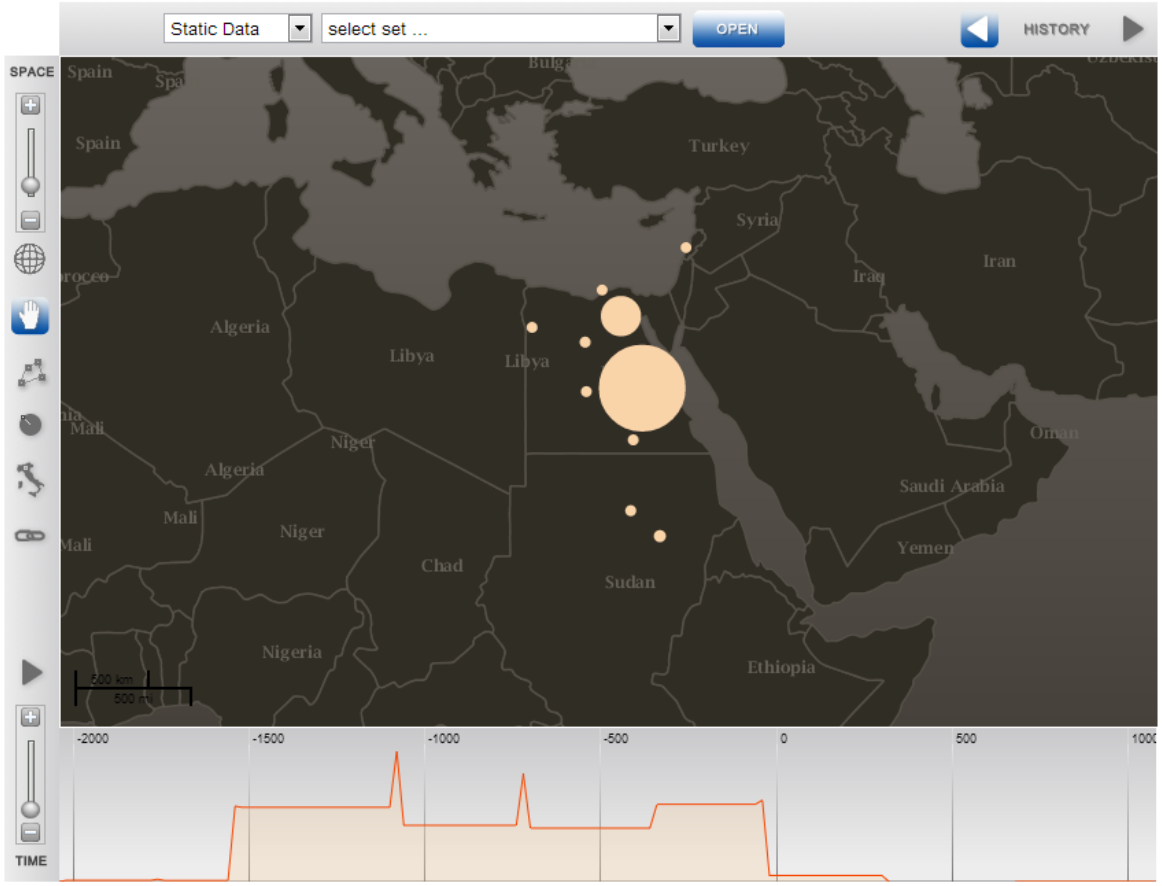


- spieldaten.zip/Kaempfer-Beispiel2/simile.html

Visualisierung?

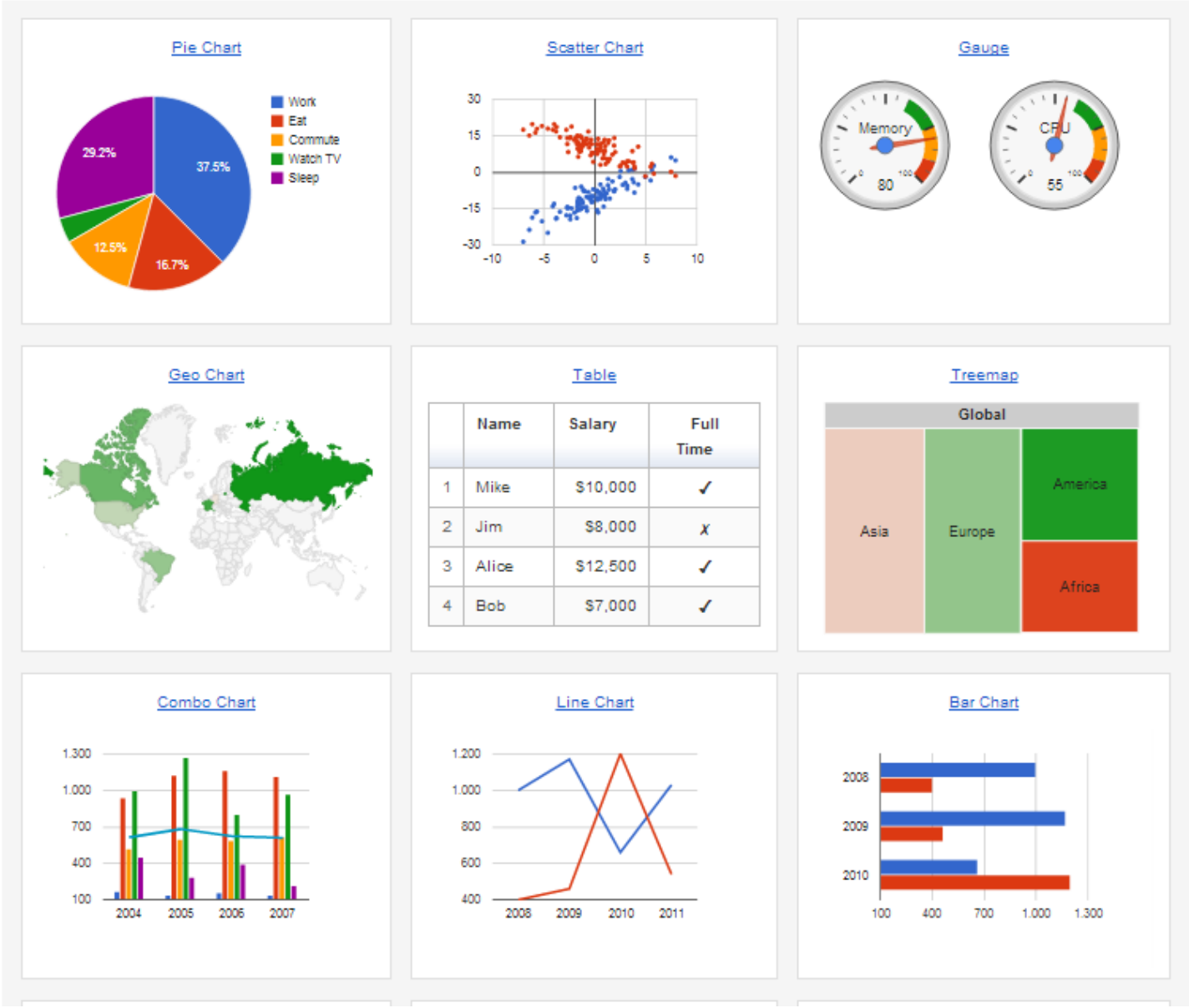


- http://www.briefedition.alfred-escher.ch (Chronologie)



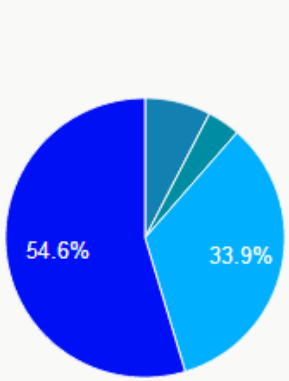
http://dev2.dariah.eu/e4d/?kml1=http%3a%2f%2fwww.cceh.uni-koeln.de%2fprojekte%2ftb%2ftotenbuch.kml&source1=1&minTime=-125533584000000&maxTime=-28432771200000&mapId=13&zoom=3.992&centerLon=3390144&centerLat=2942361.5&connections=0?kml1=http%3a%2f%2fwww.cceh.uni-koeln.de%2fprojekte%2ftb%2ftotenbuch.kml&source1=1

Google Chart API



https://developers.google.com/chart/interactive/docs/gallery

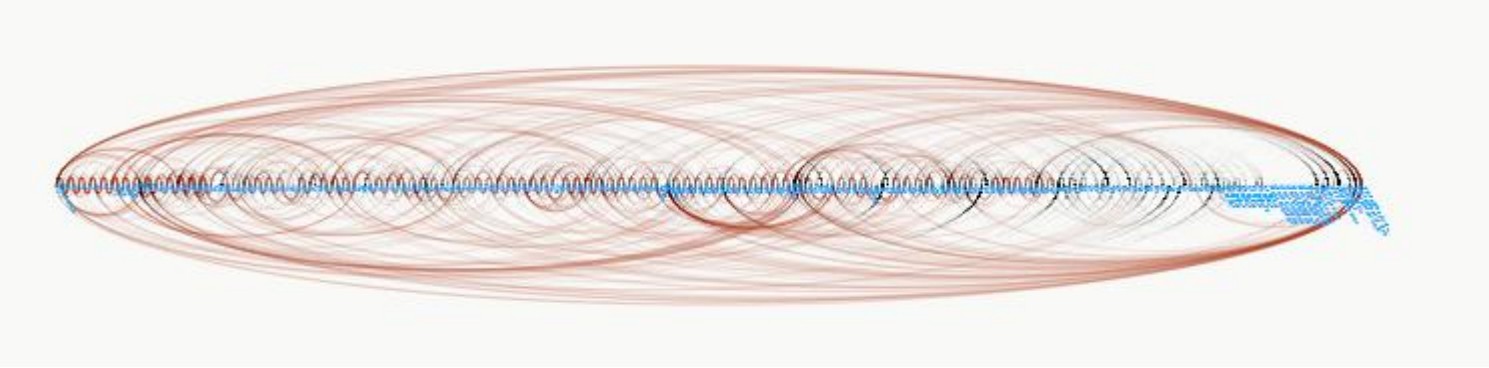
Google Charts und Maps



Region	Breiten-grad	Objekte	
Sidon	33.57	1	→
El-Sowwa	31.27	1	→
Tanis	30.97	5	→
Tell Faraun	30.88	1	→
Horbeit	30.81	1	→
Tell Abu Sefeh	30.79	1	→
Tell al Maskhuta	30.59	1	→
Athribis	30.49	1	→
Wardan	30.29	1	→
Heliopolis	30.13	3	→
El-Rizeikat	30.09	1	→
Giza	30.08	4	→
Gurob	29.92	7	→
Abusir el-Meleq	29.88	2	→
Abusir	29.88	7	→
Saqqara	29.85	191	→
Lischt	29.57	2	→
Atfih	29.41	1	→
Fayum	29.4	5	→

http://www.totenbuch.awk.nrw.de → Übersichten, Herkunft

Visualisierung per SVG



http://www.totenbuch.awk.nrw.de/static/overviews/benachbarte-sprueche-absolut-gerade-max.svg

Übung 2: Visualisierung

- Beispiel hier:
  - Geographische Karte erstellen mit Google Maps
- XSLT:
  - HTML-Grundgerüst
  - mit Javascript –Bausteinen
    - Tutorial:https://developers.google.com/maps/documentation/javascript/tutorial?hl=de
    - Lokale Einstellungen:
      - Karte: Mittelpunkt, Zoomfaktor, Funktionalitäten
      - Marker: Position, Infofenster (HTML)
  - Mit Datensätzen füllen (<xsl:for-each>/<xsl:for-each-group>)
- → Lösung (kommentiert): spieldaten.zip/Witzel/karte.xsl
- → Ergebnis: spieldaten.zip/Witzel/karte.html