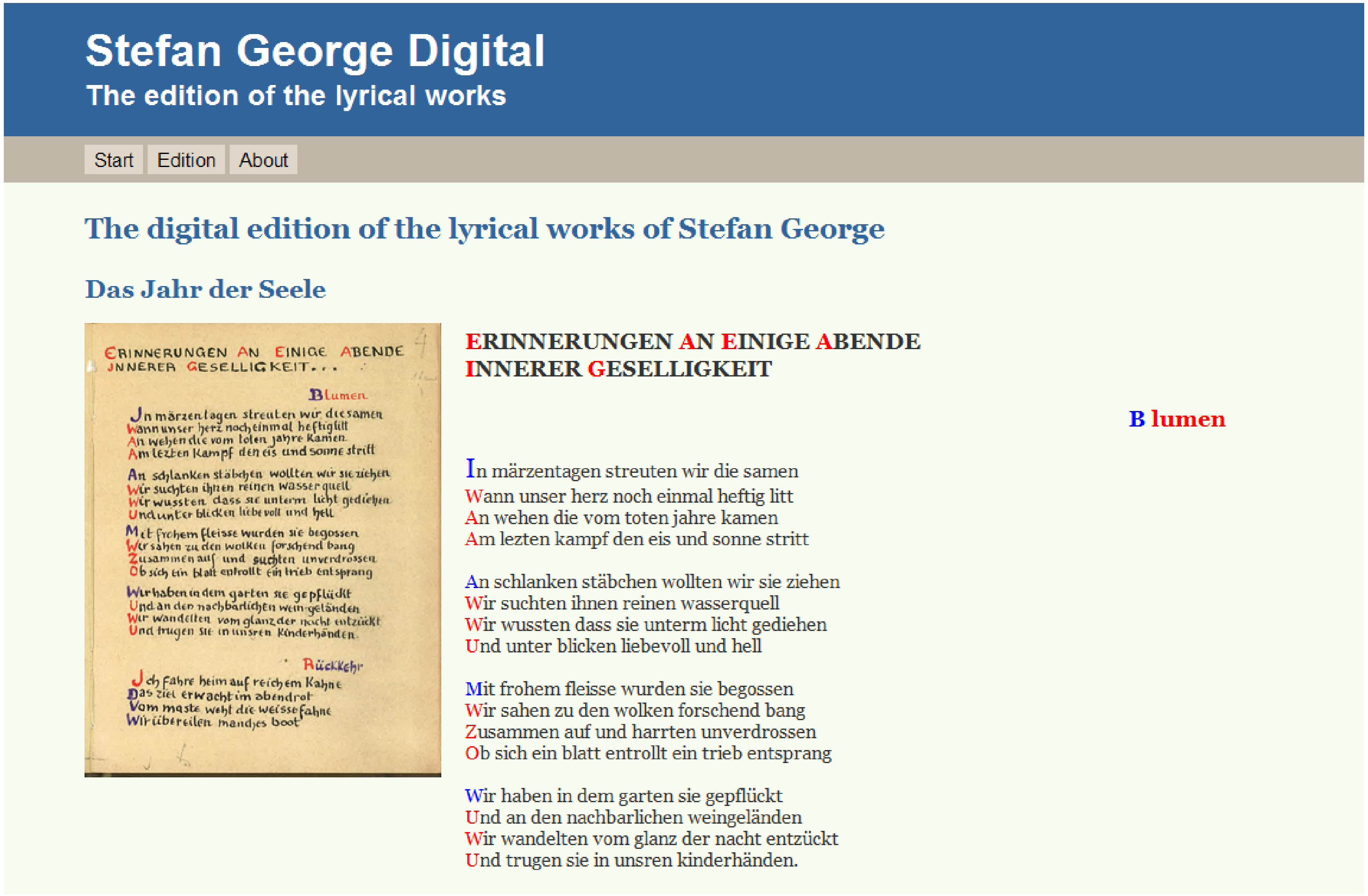
Create an HTML output from your own project with XSLT

Martina Semlak - Georg Vogeler

i|d|e

Digital Scholarly Editions Initial Training Network



Stefan George Digital

The edition of the lyrical works

The digital edition of the lyrical works of Stefan George

Das Jahr der Seele

ERINNERUNGEN AN EINIGE ABENDE INNERER GESELLIGKEIT

B lumen

In märzentagen streuten wir die samen
Wann unser herz noch einmal heftig litt
An wehen die vom toten jahre kamen
Am lezten kampf den eis und sonne stritt

An schlanken stäbchen wollten wir sie ziehen
Wir suchten ihnen reinen wasserquell
Wir wussten dass sie unterm licht gediehen
Und unter blicken liebevoll und hell

Mit frohem fleisse wurden sie begossen
Wir sahen zu den wolken forschend bang
Zusammen auf und harrten unverdrossen
Ob sich ein blatt entrollt ein trieb entsprang

Wir haben in dem garten sie gepflückt
Und an den nachbarlichen weingeländen
Wir wandelten vom glanz der nacht entzückt
Und trugen sie in unsren kinderhänden.

Minimal stuff provided

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xpath-default-namespace="http://www.tei-c.org/ns/1.0" version="2.0">
   <xsl:output method="xml"
      omit-xml-declaration="yes"
      encoding="UTF-8"
      indent="yes" />

</xsl:stylesheet>
```

First step

- Grab file
- xsl:template with the attribute: match="/"

- This template can contain the basic HTML structure of the ouput file

HTML basics – Reminder

-
  ```
  <html>
          <head><title></title><link (e.g. for a css)></head>
          <body></body>
  </html>
  ```

- We added this into
  ```
  <xsl:template match="/">

  </xsl:template>
  ```

Display text

- We have prepared basic HTML structure for you:
  header, section, nav > ul = navigation with references to other files

- We need a heading for the whole text:
  - html: section > h3
  - xsl: xsl:value-of, attribute select with the appropriate
  - xpath: //body/head

- div seems to be convenient to retain:
  - do something with all divs:

    ```
    <xsl:apply-templates select="//body/div"/>
    ```
  - what to do:

    ```
      <xsl:template match="div">
            <div><xsl:apply-templates /></div>
    </xsl:template>
    ```

XPath conditions

- Use a condition in the XPath (square brackets) for
  - html: body/header > h1 and h2:
  - find the appropriate **title**-element with **type** *main* or *sub*

```
<h1>
        <xsl:value-of select="//title[@type='main']"/>
</h1>
```

<apply-templates />: individual templates

- xsl:template match="lg"
  div class="stanza"
- xsl:template match="l"
  <br/>...
- xsl:template match="hi"
  span
- xsl:template match="lb"
  <br />

Attributes

- Convert TEI content into an attribute of any HTML-element:
- as an explicit XSL-statement:
  <xsl:attribute name="*attribute-name*">*content*</xsl:attribute>
- or with curly brackets **{ }:**
  <a href="**{***XPath***}**">...</a>
- e.g.: xsl:template match="hi"
  span and put the content of @rend into the style attribute

Attributes

```
<xsl:template match="hi">
        <span>
            <xsl:attribute name="class" select="@rend"/>
            <xsl:apply-templates />
        </span>
</xsl:template>
```

images: more complex XPath

- xsl:template match="pb"
  - html:<img>
  - add **XPath**: //surface[@xml:id=current()/@facs/substring-after(.,'#')]/graphic[1]/@url

  - look for a **surface** element: //surface
  - which fullfills a **condition** **[...]** which is
  - the **xml:id** of the surface has to be the same as the text in the **facs**-attribute of the **current** pb-element after the **hash:sign** @xml:id=current()/@facs/substring-after(.,'#')
  - and use ./graphic[1]
  - and its URL: ./@url

images: more complex XPath

```
<xsl:template match="pb">
        <img width="300">
            <xsl:attribute name="src"
select="//surface[@xml:id=current()/@facs/substring-
after(.,'#')]/graphic[1]/@url">
            </xsl:attribute>
        </img>
</xsl:template>
```

Headings

- HTML has explicit numbers for the hierarchical level of headings: h1, h2, h3, h4. Headings in the text are on level 4 onwards.
- we need to create an element with a name dependent from the current position
  - instead of writing directly an HTML element you can use xsl:element with the attribute name
  - the name is constructed by the number of divs in which the current element is nested: count(ancestor::div)

Headings

```
<xsl:template match="head">
        <xsl:element name="h{count(ancestor::div) + 3}">
           <xsl:apply-templates/>
        </xsl:element>
 </xsl:template>
```

Poem Headings

- are rightbound
- are **head** elements being children of a **division** from the **type** *poem*
  head[parent::div/@type='poem']

Conditions with xsl:if

-
  ```
  <xsl:if test=".[parent::div/@type='poem']">
        <xsl:attribute name="class">
              <xsl:text>right</xsl:text>
        </xsl:attibute>
  </xsl:if>
  ```

The template:

```
<xsl:template match="head">
        <xsl:element name="h{count(ancestor::div) + 3}">
            <xsl:if test=".[parent::div/@type='poem']">
                    <xsl:attribute name="class">
                        <xsl:text>right</xsl:text>
                    </xsl:attibute>
            </xsl:if>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>
```

Repeat something: e.g. keywords

- All terms in the keywords section should be converted into meta-elements in the html header:
- html: <meta name="keywords" content="..."/>
- Xpath: /TEI/teiHeader/profileDesc/textClass/keywords/term (or //textClass/keywords/term)
- output: **<xsl:for-each select="..."> ... </xsl:for-each>**
  - Put the XPath form above into the select attribute
  - Put the html into the xsl:foreach
  - add current() or . in curly brackets in the content-attribute of the meta element

Repeat something: e.g. keywords

```
<meta name="keywords">
    <xsl:attribute name="content">
        <xsl:for-each select="//textClass/keywords/term">
            <xsl:value-of select="."/>
         </xsl:for-each>
    </xsl:attribute>
</meta>
```

Output by position

```
<meta name="keywords">
    <xsl:attribute name="content">
        <xsl:for-each select="//textClass/keywords/term">
             <xsl:value-of select="."/>
                 <xsl:if test="position()!=last()">
                     <xsl:text>, </xsl:text>
                 </xsl:if>
         </xsl:for-each>
    </xsl:attribute>
</meta>
```

i|d|e

Digital Scholarly Editions Initial Training Network

CSS modification

- Since we don‘t have a page structure in the HTML we need additional CSS selectors:

```
div.stanza, h5 {
    margin-left: 320px
}

h6.right {
    font-size: 1.2em;
    text-align: right;
    margin-top: 1em;
}

img {
    float: left;
    padding-top: 1em
}
```

Convert overlapping markup

e.g. display text page by page:

- use the stylesheets from Sebastian Rahtz (Lyon 2011, example 8 in the zip folder): http://tei.oucs.ox.ac.uk/Talks/2011-05-26-lyon
- or James Cummings: https://github.com/jamescummings/LEAP-XSLT/blob/master/LEAP-processpb.xsl