Übung 3: XSL-Stylesheet für eine Postkarte

Ordner

A. Lege einen Ordner “XSLT-Übung3” mit den Unterordnern “styles” und “images” an
B. Das Quelldokument “Übung_Postkarte.xml” herunterladen und in den Ordner “XSLT-Übung3” speichern
C. Speichere die heruntergeladenen Digitalisate (0020r.jpg, 0020v.jpg) der Postkarte im Ordner “images”
D. Öffne das Quelldokument “Übung_Postkarte.xml”

XSLT Stylesheet

A. Öffne ein neues xsl-Dokument im Oxygen Editor. Wie bereits in der ersten Übung müssen bestimmte Informationen eingegeben werden:
   - XML-Deklaration mit @version und @encoding
   - Das öffnende und schließende `<xsl-stylesheet>` Element mit dem XSL-Namensraum (`xmlns:xsl="http://www.w3.org/1999/XSL/Transform"`) und der Version
   - Angabe des TEI-Namensraums `xmlns:tei="http://www.tei-c.org/ns/1.0"` muss hinzugefügt werden
   - Das Element `<xsl-template @match>` für den Wurzelknoten (root node)

B. Speichere die Datei im selben Ordner wie das TEI Dokument unter “Übung3.xsl”

Das Ergebnis der Transformation soll ein HTML-Dokument werden. Dafür müssen Elemente hinzugefügt werden:

C. Setze den Curser zwischen das öffnende und schließende Element `<xsl:stylesheet> </xsl:stylesheet>`. Tippe eine öffnende Spitzklammer ein und wähle das Element `<output>`. Füge die Attribute `@indent` mit dem Wert `yes` und `@method` mit dem Wert `html` ein:
   `<xsl:output indent="yes" method="html"/>`

Mit der `xsl:output` Deklaration wird angegeben, wie der Ergebnisbaum ausgegeben bzw. geschrieben werden soll.
Das Attribut `@indent` (optional) kann die Werte `yes` oder `no` haben. Bei `yes` wird der Ergebnisbaum so formatiert, dass untergeordnete Elemente weiter eingerückt werden. Dies hat keine Auswirkung auf die Darstellung, sondern ist eine "kosmetische" Angabe zur besseren Lesbarkeit des erzeugten Quelltextes. Die Voreinstellung für dieses Attributs ist bei `method="html"` yes, bei `method="xml"` no.
Das Attribut `@method` (optional) gibt an, nach welcher Art der Ergebnisbaum erzeugt werden soll. Mögliche Werte sind bspw. xml, html oder text.

D. Jedes HTML Dokument hat als Wurzelelement das Element `<html>`. Diese kann entweder mit `<xsl:element @name>` generiert werden oder man kann es einfach als Element in das XSLT

Stylesheet schreiben.
Schreibe das öffnende und schließende html-Tag `<html> </html>` in das `<xsl:template>`.

E. Verschachtle den Kopf `<head>` und den Titel `<title>` des HTML-Dokumentes im `<html>`-Wurzelelement. Das Stylesheet sollte nun folgendermaßen aussehen:

```
<?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:output indent="yes" method="html"/>
    <xsl:template match="/">
        <html>
            <head>
                <title/>
            </head>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

F. Datei speichern

G. Man könnte nun den Titel des HTML-Dokumentes händisch eintippen. z.B.:

```
<title>Graz-Bismarckplatz</title>
```

H. Andererseits kann der Titel auch dynamisch aus dem XML-Quelldokument ausgelesen und an der Stelle eingefügt werden, an der wir die Information brauchen. Nämlich innerhalb des `<title>`-Elementes:

```
<title>
      <xsl:value-of select="tei:TEI/tei:teiHeader/
      tei:fileDesc/tei:titleStmt/tei:title[@type='main']" />
</title>
```

I. Datei speichern

Transformationsszenario definieren

Transformationsszenario wie in Übung 1 konfigurieren. Das HTML Dokument speichern wir im Ordner wo auch das XML Dokument liegt.
Wenn wir das Ergebnis im Browser anzeigen lassen, ist noch nicht viel sichtbar. Das Ergebnis zeigt nur die Bezeichnung des HTML-Dokumentes.

Einfügen der Metadaten

A. Nun fügen wir ein `<body>` Element, eine Überschrift `<h1>` und ein `<div class='meta'>` Element für die Metadaten aus dem `<tei:teiHeader>` im `<body>` hinzu.

Das Stylesheet sollte nun wie folgt aussehen:

```
<?xml version ="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:tei="http://www.tei-c.org/ns/1.0">
<xsl:template match="/">
      <html>
         <head>
            <title><xsl:value-of
                  select="/tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/tei:tit
            le[@type='main']"/></title>
         </head>
         <body>
            <h1> </h1>
            <div class='meta'></div>
         </body>
      </html>
</xsl:template>
</xsl:stylesheet>
```

B. Die Überschrift soll gleich lauten, wie die Bezeichnung des HTML-Dokumentes. Dafür muss wieder in `<xsl:value-of @select>` der Pfad angegeben werden, der zum Titel im `<tei:teiHeader>` des Quelldokumentes führt.

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

C. Die Metadaten zu der erfassten Postkarte, die im Quelldokument unter `<fileDesc>` erfasst sind, sollen in `<div class='meta'>` eingetragen werden:

```
<div class='meta'>
      <xsl:apply-templates select="/tei:TEI/tei:teiHeader/tei:fileDesc"/>
</div>
```

Das Dokument sollte nun wie folgt aussehen:

```
<?xml version ="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:tei="http://www.tei-c.org/ns/1.0">
   <xsl:template match="/">
      <html>
        <head>
           <title>
              <xsl:value-of select="/tei:TEI/tei:teiHeader/tei:fileDesc/
                              tei:titleStmt/tei:title[@type='main']"/>
            </title>
```

```
        </head>
        <body>
           <h1>
            <xsl:value-of select="/tei:TEI/tei:teiHeader/tei:fileDesc/
                        tei:titleStmt/tei:title[@type='main']"/>
           </h1>
           <div class='meta'>
              <xsl:apply-templates
               select="/tei:TEI/tei:teiHeader/tei:fileDesc"/>
            </div>
        </body>
      </html>
 </xsl:template>
</xsl:stylesheet>
```

D. Starte die bereits konfigurierte Transformation. Das Ergebnis sollte etwa so aussehen:

Graz-Bismarckplatz

Graz-Bismarckplatz GrazMuseum Postkartensammlung Online Unbekannt Transcription Carina Koch Zentrum für Informationsmodellierung, Universität Graz GrazMuseum Graz Sackstraße 18, A-8010 Graz Postkarte ASK 05/00020

Strukturieren der Metadaten

Das Ergebnis ist noch etwas unschön. Nun wollen wir es strukturierter gestalten.

A. Zuerst füge den Untertitel des Quelldokumentes unter dem Titel (`<h1>`) in einem `<h2>` Element hinzu.
B. Versuche nun die Informationen im `<div>` Element zu strukturieren. Obwohl nur das Element `<tei:fileDesc>` ausgewählt wurde, werden alle Informationen dazu angezeigt. Das liegt daran, weil der XSL Prozessor für jedes Element eine default Schablone benutzt und dadurch alles angezeigt wird. Nun wollen wir Schablonen für einige ausgewählt Elemente definieren.
    a. Statt alle Elemente mit `<xsl:apply-templates select="/tei:TEI/tei:teiHeader/tei:fileDesc" />` anzusprechen, nutze `<xsl:text>, <xsl:value-of>` und HTML Elemente (z.B.: `<br/>` für Zeilenumbrüche). Das `<xsl:text>` Element wird für Text und Leerzeichen genutzt.

```
<div class="meta">
      <xsl:text>Titel: </xsl:text>
      <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/
```

```
                        tei:titleStmt/tei:title[@type='main']"/>
      <br/>
      <xsl:text>Quelle: </xsl:text>
      <xsl:value-of select="tei:TEI/tei:teiHeader/
                        tei:fileDesc/tei:sourceDesc/tei:p"/>
</div>
```

C. Starte die bereits konfigurierte Transformation. Das Ergebnis könnte wie folgt aussehen:

Hier werden nicht alle Informationen aus dem Quelldokument ausgelesen, sondern nur die Informationen, die gewünscht sind und ausgewählt wurden. Auch die Informationen in `<respStmt>` und `<publicationStmt>` könnte so ausgegeben werden.

Ausgabe des Postkartentexts

Nachdem wir nun die Metainformationen zu der Quelle anzeigen, widmen wir uns im nächsten Schritt der Darstellung der Transkription der Postkarte.

A. Der Textinhalt ist in Kindknoten des Knoten `<tei:text>` und `<tei:body>` gespeichert. Um darauf Zugriff zu bekommen wird zunächst der Knoten `<tei:text>` und `<tei:body>` an der Stelle aufgerufen, an der der Textinhalt eingefügt werden soll (hier nach den Metainformationen unter
`<div class=”meta”></div>`.

```
<xsl:apply-templates select="tei:TEI/tei:text/tei:body"/>
```

Das Dokument sollte jetzt wie folgt aussehen:

```
<?xml version ="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
xmlns:tei="http://www.tei-c.org/ns/1.0">

<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <title>
```

```
            <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/
                                  tei:titleStmt/tei:title[@type = 'main']"/>
         </title>
      </head>
      <body>
         <h1>
            <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/
                                  tei:title[@type = 'main']"/>
         </h1>
         <h2>
            <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/
                                  tei:title[@type = 'sub']"/>
         </h2>
         <div class="meta">
            <xsl:text>Titel: </xsl:text>
            <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/
                                  tei:title[@type = 'main']"/>
         <br/>
            <xsl:text>Quelle: </xsl:text>
            <xsl:value-of
            select="tei:TEI/tei:teiHeader/tei:fileDesc/
                                  tei:sourceDesc/tei:p "/>
         </div>
         <xsl:apply-templates select="tei:TEI/tei:text/tei:body"/>
      </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

B. Weiters wollen wir die Kindelemente von `tei:text/tei:body` aufrufen. Dafür öffnen wir direkt unter dem schließenden `</xsl:template>` ein neues `<xsl:template>`, das `tei:text/tei:body` entspricht und indem seine Kindelemente aufgerufen werden:

```
<xsl:template match="tei:div">
      <xsl:apply-templates select="tei:opener"/>
      <xsl:apply-templates select="tei:p"/>
      <xsl:apply-templates select="tei:closer"/>
</xsl:template>
```

C. Nun sollen drei Templates definiert werden
`<tei:opener>`, `<tei:p>`, `<tei:closer>`.

```
<xsl:template match="tei:opener">
      <p><span>
            <xsl:text>Anrede: </xsl:text>
      </span>
      <span>
            <xsl:apply-templates select="tei:salute"/>
      </span></p>
</xsl:template>
```

```
<xsl:template match="tei:p">
	<p><span>
		<xsl:text>Mitteilung: </xsl:text>
	</span>
	<span>
		<xsl:apply-templates></xsl:apply-templates>
	</span></p>
</xsl:template>

<xsl:template match="tei:closer">
	<p><span>
		<xsl:text>Grußformel: </xsl:text>
	</span>
	<span>
		<xsl:apply-templates select="tei:salute"/>
	</span></p>
</xsl:template>
```

Hier wurde die Syntax `@select="."` genutzt. Dabei handelt es sich um eine Abkürzung, die auf den Knoten verweist, der gerade angesprochen ist (in der XSL Syntax `"self::node()"`. In dem Fall auf `<tei:p>`.

Einbinden eines Hyperlink

Der Ortsname 'Wagstadt' ist Georeferenziert über geonames.org. Das `@ref` Attribut enthält eine URI, die den Ortsnamen identifiziert. Mit XSLT wollen wir nun einen Hyperlink setzen, der auf die geonames.org Seite weiterführt.

A. Zuerst wollen wir ein Template für das `<tei:placeName>` Element schaffen. Füge folgendes Template zu deinem XSLT.

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

B. Starte die bereits konfigurierte Transformation. Das Ergebnis sollte so aussehen wie zuvor.
C. Nun legen wir einen Hyperlink um den Ortsnamen. Dafür benützen wir das Element `<xsl:element>` und geben ihm das Attribut `@name` mit dem Wert `'a'`:

```
<xsl:template match="tei:placeName">
	<xsl:element name="a">
		<xsl:apply-templates/>
	</xsl:element>
</xsl:template>
```

D. Zusätzlich zum HTML Element `<a>` braucht ein Hyperlink auch ein Attribut `@href` an diesem Element. Das Attribut `@href` enthält die URL des Zieldokuments des Hyperlinks. In unserem Beispiel kann diese URL aus dem Attribut `@ref` des `<tei:placeName>`

Elements generiert werden. Um das Attribut `@href` im HTML zu generieren verwenden wir den `<xsl:attribut>` Befehl.

```
<xsl:template match="tei:placeName">
      <xsl:element name="a">
                  <xsl:attribute name="href" select="@ref">
                  </xsl:attribute>
                  <xsl:apply-templates></xsl:apply-templates>
      </xsl:element>
</xsl:template>
```

E. Starte die bereits konfigurierte Transformation und du solltest nun einen funktionierenden Hyperlink in der Transkription haben.

Einbinden eines Bildes

Zunächst muss im `<body>`, an der Stelle, wo die Bilder angezeigt werden sollen, auf den Pfad im Quelldokument verwiesen werden:

```
<xsl:apply-templates select="tei:TEI/tei:facsimile"/>
```

Auch für das Einbinden der Bilder wird ein Template benötigt:

```
<xsl:template match="tei:graphic">
        <img width="300px">
            <xsl:attribute name="src">
                <xsl:value-of select="@url"/>
            </xsl:attribute>
        </img>
</xsl:template>
```

HTML `<img>` Tag: https://www.w3schools.com/TagS/tag_img.asp

Das XSL-Dokument sollte final so aufgebaut sein:

```
<?xml version ="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
    xmlns:tei="http://www.tei-c.org/ns/1.0">
    <xsl:template match="/">

        <html xmlns="http://www.w3.org/1999/xhtml">
              <head>
                <title>
                    <xsl:value-of
                  select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/tei:titl
                  e[@type = 'main']"/>
                </title>
            </head>
            <body>
                <h1>
                    <xsl:value-of
```

```
                select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/tei:titl
                e[@type = 'main']"/>
            </h1>
            <h2>
                <xsl:value-of
                select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/tei:titl
                e[@type = 'sub']"/>
            </h2>
            <div class="meta">
                <xsl:text>Titel: </xsl:text>
                <xsl:value-of
                select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:titleStmt/tei:titl
                e[@type = 'main']"/>
                <br/>
                <xsl:text>Quelle: </xsl:text>
                <xsl:value-of
                select="tei:TEI/tei:teiHeader/tei:fileDesc/tei:sourceDesc/tei:p"/
                >
            </div>
            <xsl:apply-templates select="tei:TEI/tei:text/tei:body"/>
            <xsl:apply-templates select="tei:TEI/tei:facsimile"/>
        </body>
    </html>
</xsl:template>
<xsl:template match="tei:div">
    <xsl:apply-templates select="tei:opener"/>
    <xsl:apply-templates select="tei:p"/>
    <xsl:apply-templates select="tei:closer"/>
</xsl:template>
<xsl:template match="tei:opener">
    <p xmlns="http://www.w3.org/1999/xhtml">
        <span>
            <xsl:text>Anrede: </xsl:text>
        </span>
        <span>
            <xsl:value-of select="tei:salute"/>
        </span>
    </p>
</xsl:template>
<xsl:template match="tei:p">
    <p xmlns="http://www.w3.org/1999/xhtml">
        <span>
            <xsl:text>Mitteilung: </xsl:text>
        </span>
        <span>
            <xsl:value-of select="."/>
        </span>
    </p>
</xsl:template>
<xsl:template match="tei:closer">
    <p xmlns="http://www.w3.org/1999/xhtml">
```

```
            <span>
                <xsl:text>Grußformel: </xsl:text>
            </span>
            <span>
                <xsl:value-of select="tei:salute"/>
            </span>
        </p>
    </xsl:template>
    <xsl:template match="tei:graphic">
        <img xmlns="http://www.w3.org/1999/xhtml" width="300">
            <xsl:attribute name="src">
                <xsl:value-of select="@url"/>
            </xsl:attribute>
        </img>
    </xsl:template>
</xsl:stylesheet>
```

Weiterführende Übung CSS:

Neben HTML-Tags kann mit Cascading Style Sheets (CSS) die Ausgabe weiter gestaltet werden.

Es gibt 3 Möglichkeiten HTML und CSS zu verknüpfen:

1. direkt im Quellcode:

```
<h1 style='font-style:italic;'>      </h1>
<h1 style='font-style:italic; color: red;'>      </h1>
```

2. am Anfang der HTML-Datei

```
<head>
<style>
      h1 {
      font-style:italic;
      color: red;
      }
</style>
</head>
```

3. ausgelagert in extra CSS-Datei

Zum Erlernen ist die zweite Variante am geschicktesten. Dort sind die CSS-Anweisungen direkt in der HTML-Datei eingebunden und es kann schnell getestet werden.
Für die Entwicklung von Websites empfiehlt sich dann die 3. Variante. Dort werden die CSS-Anweisungen in einer externen Datei hinterlegt und diese Datei wird dann in jede HTML-Seite eingebunden (= vorbereitet).
Lade die Datei Übung3.css herunter, speichere sie im Ordner “XSLT-Übung3” und binde sie in dein Stylesheet ein.

```
<xsl:template match="/">
<html>
```

```
  <head>
      <title>
            <xsl:value-of select="tei:TEI/tei:teiHeader/tei:fileDesc/
                                  tei:titleStmt/tei:title[@type = 'main']"/>
      </title>
      <link rel="stylesheet" href="Übung3.css"/>
…
```

Eine Liste von CSS Selektoren findet ihr hier: https://www.w3schools.com/cssref/css_selectors.asp
Eine Liste von CSS Eigenschaften und Möglichkeiten findet ihr hier:
https://www.w3schools.com/cssref/default.asp

Verändere die Textwerte in den `<span>`-Elementen.
Um unterschiedliche Formatierungen anwenden zu können müssen die `<span>`-Elemente unterscheidbar werden. Dafür kann man das Attribut `@class` nutzen. Schreibe dein XSLT um und füge ein Attribut `class` an folgende Elemente:

```
<xsl:template match="tei:closer">
      <p><span class="category">
            <xsl:text>Grußformel: </xsl:text>
      </span>
      <span class="transcription">
            <xsl:value-of select="tei:salute"/>
      </span> </p>
</xsl:template>
```

```
Bsp: Anweisung in der CSS Datei:
.category{
font-style: italic;
font-size: 120%;
color: blue;
}
.transcription{
font-family: Georgia, Serif;
color: green;
}
```

Die Klassenwerte können in CSS über den “.” (Punkt-Selector) aufgerufen werden.
Siehe dazu: CSS Selektoren https://www.w3schools.com/cssref/css_selectors.asp

Als nächstes versuche die Metadaten rot zu setzen. Dafür muss der Klassenwert ‘meta’ angesprochen werden.