EVT User Manual

For EVT v. 1.2

Introduction

This guide is aimed at the editor interested in creating an image-based edition using EVT, and not at the reader/user of such edition. To use EVT, the editor must have a good knowledge of the TEI XML markup, and must be able to apply XSLT style sheets to an XML document. EVT main configuration is done by editing a single .xsl file (see section *3. Configuring EVT*), note however that the software can be customized as desired modifying the relevant XSLT and CSS style sheets, or adding new ones (f.i. for a different kind of text visualization).

About EVT

EVT (Edition Visualization Technology) is a software for creating and browsing digital editions of manuscripts based on text encoded according to the TEI XML schemas and Guidelines. This tool was born as part of the DVB (Digital Vercelli Book) project in order to allow the creation of a digital edition of the Vercelli Book, a parchment codex of the late tenth century, now preserved in the Archivio e Biblioteca Capitolare of Vercelli and regarded as one of the four most important manuscripts of the Anglo-Saxon period as regards the transmission of poetic texts in the Old English language.

To ensure that it will be working on all the most recent web browsers, and for as long as possible on the World Wide Web itself, the edition-browsing component is built on open and standard web technologies such as HTML, CSS and JavaScript. Specific features, such as the magnifying lens, are entrusted to jQuery plugins, again chosen among the open source and best supported ones to reduce the risk of future incompatibilities. The general architecture of the software, in any case, is modular, so that any component which may cause trouble or turn out to be not completely up to the task can be replaced easily.

How it Works

The basic idea of EVT is very similar to the *modus operandi* which is commonly used to convert TEI XML documents into HTML: when the main style-sheet is applied to the document, it starts a processing which ends with a website containing the digital edition of the manuscript. Our ideal goal, in fact, is to have a simple, very user-friendly drop-in tool, requiring little work and/or knowledge of anything beyond XML from the editor. To reach this goal, EVT is based on a modular structure where a single style sheet (evt_builder.xsl) starts a chain of XSLT 2.0 transformations calling in turn all the other modules; the latter belong to two general categories: those devoted to building the HTML site, and the XML processing ones, which extract the edition text (the content of each folio which lies between <pb/> elements, or the content of each <surface> when the Embedded Transcription method is used) and format it according to the edition level. All

XSLT modules live inside the builder_pack folder, in order to have a clean and well organized directory hierarchy.

Main Features

At the present moment EVT can be used to create image-based editions with two possible edition levels: diplomatic and diplomatic-interpretative. This means that a transcription encoded using elements of the TEI ***transcr*** module (see chapter 11 *Representation of Primary Sources* in the TEI *Guidelines*) should already be compatible with EVT, or require only minor changes to be made compatible. The Vercelli Book transcription is following the standard TEI schemas with no custom elements or attributes added: our tests with similarly encoded texts showed a high grade of compatibility. A critical edition level is being studied and will be added to the next version of EVT.

On the image side, several features such as a magnifying lens, a general zoom, image-text linking and more are already available. The image-text feature is inspired by Martin Holmes' Image Markup Tool software and was implemented in XSLT and CSS by one of the students collaborating to the project; all other features are achieved by using jQuery plugins.

A Quick Guide to EVT

To create an image-based edition EVT requires both images and TEI XML documents as input data. For the XSLT style sheets to work properly, all the input data must follow specific rules:

- images have to be named according to a strict naming convention and have to be available in a few different sizes/resolutions;
- TEI XML documents are likewise bound not only by the encoding schema, which must include the ***transcr*** module, but also by the use of several attributes whose values must follow rigid conventions.

All these conventions, which will be explained in full detail in the following sections, are necessary because they build a layer of rich metadata information which is accessed by the different XSLT style sheets to find all the relevant data to be extracted, processed and finally presented to the user.

After preparing all the image and textual data according to the conventions that will be explained below (sections *1. Images* and *2. TEI XML Transcription Files*), the user will be required to check and if necessary modify the EVT configuration file to suit her/his needs (see section *3. Configuring EVT*), and finally follow a step-by-step path to put everything in place and start the transformation chain that will result in a web-based edition (see section *4. Step by Step Instructions*).

This guide tries to include examples for each case described, however remember that the zipped archive available on SourceForge includes all text and image files necessary for the sample editions it holds, so that you can inspect them in case of doubt. In the `builder_pack/doc` folder you will find a first draft of the technical documentation, automatically generated thanks to the XSLTdoc system.

The main file for the transformation is `evt_builder.xsl`, which calls all the other `evt_builder-*.xsl` modules using XInclude.

As a first step of the process, the XSL style sheets that we consider “basic templates” are called into action: these are used to generate the index page in HTML and the different outputs depending on the edition levels specified by the editor. This is also the phase where all the parameters and variables that we defined in the configuration file are used by the transformation chain, and the structure for the page-by-page navigation is generated as well.

After the structural generation part is over, it is now the turn of the files designed for the implementation of specific features, such as text-image linking or the magnifying lens. The style sheets in the `modules/elements` folder are those defining the visual aspect of the edition text for the different edition levels.

0. Installation

EVT installation is quite easy: you only have to download the zipped archive from SourceForge and unpack it in some suitable folder on your hard disk. Everything is already in place, first of all you should browse the included examples clicking on the index.html document. You can also browse the available documents in the data/input_data folder, especially the TEI XML files, to understand what the building system requires; if you feel particularly adventurous you’re welcome to browse the XSLT style sheets in the `builder_pack` and `builder_pack/modules` folders, note however that you should try to modify these files only if you know what you are doing!

The data/input_data folder is where your own data will go before you can use EVT to produce a web edition of your manuscript.

1. Images

As of version 1.1 EVT supports digitized images of the manuscript(s) saved in any of the most popular standards: JPEG, PNG, TIFF etc. Note, however, that

1. the actual image format must be stated using the corresponding extension in the main configuration file (see section *3. Configuring EVT*), e.g. `<xsl:param name="imageExt">jpg</xsl:param>`
2. all images used in an edition must belong to the same format: if you choose JPEG, all of your images must be JPEGs and have the .jpg extension, the same is true for PNG, etc.

We suggest using one of the most popular and effective standards, such as PNG or JPEG, to reduce loading times in the web-edition.

Images should be provided in three different types:

- single side images
- double side images
- detail (hot spot) images

and the first two types must be made available in three different resolutions:

- standard
- high (mandatory for the Magnifier tool to work properly)
- thumbnail (not yet used for double side images)

Note that, although only single side images are actually necessary, it is recommended to provide double side ones too, so that the Bookreader view is available to browse the manuscript showing verso-recto folios. If for some reason the double side images aren't available, however, the Bookreader view can be disabled in the configuration file (see below).

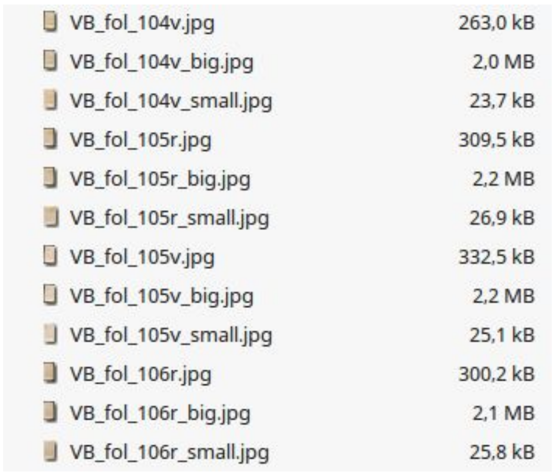
1.1 Single Side Images

These are single page/folio images of the manuscript:

- **standard** resolution: each image must be named "[folio_number].jpg" (e.g. "104v.jpg"), where "[folio_number]" is the value of the `@xml:id` attribute of the `<pb/>` element (if using the Parallel Transcription method) or the `<surface>` element (if using the Embedded Transcription method) that is used to encode the corresponding folio;
- **high** resolution images: these are necessary for the magnifier (the proportion must be the same as the normal ones) and are named "[folio_number]_big.jpg" (e.g. "104v_big.jpg"), where "[folio_number]" is the name of the corresponding image in standard resolution (note the "**_big**" suffix);
- **thumbnails** of the standard images must be named "[folio_number]_small.jpg" (e.g. "104v_small.jpg"), where "[folio_number]" is the name of the corresponding image (note the "**_small**" suffix).

Important: note that it is recommended to use a meaningful prefix including, f.i., a reference to the manuscript (e.g. "VB_fol_104v", which means "folio 104v of the Vercelli Book manuscript"), useful when dealing with more than one manuscript; since at the present moment EVT can only handle one manuscript, this is not essential to its proper functioning.

Examples:



```
VB_fol_104v.jpg          263,0 kB
VB_fol_104v_big.jpg        2,0 MB
VB_fol_104v_small.jpg     23,7 kB
VB_fol_105r.jpg          309,5 kB
VB_fol_105r_big.jpg        2,2 MB
VB_fol_105r_small.jpg     26,9 kB
VB_fol_105v.jpg          332,5 kB
VB_fol_105v_big.jpg        2,2 MB
VB_fol_105v_small.jpg     25,1 kB
VB_fol_106r.jpg          300,2 kB
VB_fol_106r_big.jpg        2,1 MB
VB_fol_106r_small.jpg     25,8 kB
```

Note that you will have to find a suitable compromise with regard to image size/resolution and actual performance of the edition on the Web: bigger images may be better for detail inquiries, but they take longer to load and to navigate, especially on low speed Internet connections. The current beta of the Digital Vercelli Book is geared more on navigation speed than image detail.

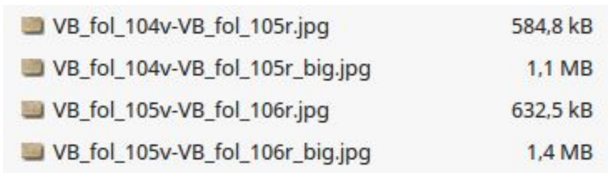
1.2 Double Side Images

These are double side images of the manuscript, showing the verso of a folio and the recto of the following one:

- **standard** resolution double folio images: these images are necessary for the functioning of the Bookreader view and must be named "[left_folio_number]-[right_folio_number].jpg" (e.g. "104v-105r.jpg"), where "[left_folio_number]" and "[right_folio_number]" are the names of images of the single facing folios, respectively the "verso" (on the left hand side) and the "recto" (on the right hand side); the double image name is therefore composed by the names assigned to the image of each single folio;
- **high** resolution double folios images: these images are necessary to activate the magnifier in the Bookreader view (the proportion must be the same as the normal ones) and must be named "[left_folio_number]-[right_folio_number]**_big.**jpg" (e.g. "104v-105r_big.jpg"), where "[left_folio_number]-[right_folio_number]" is the name of the corresponding double folio normal image.

Note that for the first and last page of the manuscript and for all the pages that do not have a "partner", it is still mandatory to have the images for the double view; however, it is sufficient that they be named as usual: [left_folio_number].jpg (e.g. “135v.jpg”) if it doesn't have the right partner, and [right_folio_number].jpg (e.g. “103r.jpg”) if it doesn't have the left partner. The same rule applies to the high resolution images, they must be named [left_folio_number]_big.jpg (e.g. “135v_big”) and [right_folio_number]_big.jpg (e.g. “103v_big.jpg”).

Examples:



VB_fol_104v-VB_fol_105r.jpg	584,8 kB
VB_fol_104v-VB_fol_105r_big.jpg	1,1 MB
VB_fol_105v-VB_fol_106r.jpg	632,5 kB
VB_fol_105v-VB_fol_106r_big.jpg	1,4 MB

1.3 Hotspot Images

These alternative images for specific areas of a manuscript folio, e.g. a more detailed version produced thanks to virtual restoration, possibly to be accompanied by a textual note. The only requirement is that they are named in the same way as they are referred to in the XML file, it is recommended that they are included in the <back> section of a text:

```
<back>
   <div type="hotspot">
     <div facs="VB_hs_106r_01" xml:id="VB_div_hs_106r_01">
       <p>The scribe here wrote a word, 'þrowode', that he
        later corrected in 'wolde' by erasing the first
        three letters and adding an 'l'. In this image you
        can see the contours of the erased letters. </p>
       <figure>
         <graphic url="VB_106r_HS_01.jpg"/>
       </figure>
     </div>
   </div>
 </back>
```

1.4 Other Images

Depending on specific project goals, it is possible that you have a set of images created for a specific purpose: in the Codice Pelavicino Digitale project, for example, all the notary graphic symbols have been collected to form a *signa tabellionis* database, and have been placed in a separate `images/signum` folder. Again this is supported by EVT, provided that the correct links to images are provided in the XML documents.

2. TEI XML Transcription Files

2.0 TEI Header

The `<teiHeader>` element at the top of the XML file must be compiled according to the standard TEI *Guidelines*, providing all the necessary information about the project and its curators. Note that if you decide to include one or more `<msDesc>` elements these will be used to show information pertaining to the corresponding manuscripts in the image frame, where the digitized manuscript images are showed.

Please also note that if you wish to take advantage of the support for named entities, this is where you should insert the lists (e.g. `<listPerson>`), this is explained in more detail in the *2.4 Support for named entities* section.

2.1 Facsimile

The `<facsimile>` element is where all the information necessary for image-text linking is saved, both for hot-spot and for continuous linking (e.g. page or line level) purposes. Note that you can do without this element in the transcription document, EVT will work and show your texts even if there are no images: to avoid having an empty image frame, though, you may want to set the `image_frame` variable in the configuration file (see below) to `false()` since EVT doesn't check the existence of this element (yet) and generates an image frame if that variable is left to `true()`.

A `<facsimile>` holds as many `<surface>`s as there are single side images. Each `<surface>` includes:

- a `@corresp` attribute pointing to the `@xml:id` of the corresponding `<pb/>` element

```
<surface xml:id="VB_surf_104v" corresp="#VB_fol_104v">
```

- a `<graphic>` element: this can include an `@url` attribute pointing to a specific file location, but at the present moment only `@height` and `@width` are used by EVT;

```
<graphic width="1200px" height="1800px"
         url="../images/Vercelli-Book_104V_S_300dpi.jpg"/>
```

- a number of `<zone>` elements: these are optionally used to record the coordinates of image areas corresponding to text elements and other metadata information, to do so they require use of several attributes:

◦ **ulx**, **uly**, **lrx**, **lry**	cartesian coordinates of the image area
◦ **rendition**	two possible values: **Line** (for line-by-line text-image linking) and **HotSpot** (for image areas to be used in hotspots)
◦ **xml:id**	the unique ID for the current `<zone>`
◦ **corresp**	points to the corresponding `<lb/>` ID (i.e. line) in the text transcription (OPTIONAL)

```
<zone corresp="#VB_lb_104v_01" lrx="1052" lry="211"
      rend="visible" rendition="Line" ulx="261" uly="156"
      xml:id="VB_line_104v_01"/>
```

Complete example:

```
<facsimile xml:id="VB_fac_dotr">
  <surface xml:id="VB_surf_104v" corresp="#VB_fol_104v">
    <graphic height="1800px" width="1200px"
             url="immagini\Vercelli-Book_104V_S_300dpi.jpg" />
    <zone corresp="#VB_lb_104v_01" lrx="1052" lry="211"
          rend="visible" rendition="Line" ulx="261" uly="156"
          xml:id="VB_line_104v_01" />
    <zone corresp="#VB_lb_104v_02" lrx="1072" lry="263"
          rend="visible" rendition="Line" ulx="257" uly="209"
          xml:id="VB_line_104v_02" />
     [...]
    <zone corresp="#VB_lb_104v_23" lrx="1084" lry="1318"
          rend="visible" rendition="Line" ulx="262" uly="1269"
          xml:id="VB_line_104v_23" />
    <zone corresp="#VB_lb_104v_24" lrx="1104" lry="1372"
          end="visible" rendition="Line" ulx="260" uly="1316"
          xml:id="VB_line_104v_24" />
  </surface>
```

For more information about digital facsimiles in TEI XML see chapter 11 Representation of Primary Sources of the *Guidelines*.

2.2 Parallel Transcription

The Parallel Transcription method is the most popular and recommended way to couple a (semi-)diplomatic transcription with digitized manuscript images. This is what EVT expects to find in a TEI XML document created according to this method:

- **`<text>`** follows the **`<facsimile>`** element and holds the following items:

    - `<front>` front matter (information about a text, regesto (diplomatic text summary), etc.)
    - `<body>` actual text
    - `<back>` back matter (notes, comments, etc.)

- EVT also support the **<group>** element, so that you can have more than one <text> in a single TEI document, which is perfect for miscellaneous manuscripts:

  ```
  <text>
      <front> [ front matter for the whole document ]  </front>
      <group>
          <text>
              <front>  [ front matter for the first text ]  </front>
              <body>  [ body of the first text ]            </body>
              <back>  [ back matter for the first text ]    </back>
          </text>
          <text>
              <front>  [ front matter for the second text]  </front>
              <body>  [ body of the second text ]           </body>
              <back>  [ back matter for the second text ]   </back>
          </text>
          ... [ more texts ] ...
      </group>
      <back> [ back matter for the whole document ]  </back>
  </text>
  ```

  Note that you can also XInclude to keep the texts in separate TEI documents.

- To provide a safe starting point for the chain of XSLT transformations it is essential that each `<text>` includes an **xml:id** attribute holding an unique ID for the text. Also, while not mandatory, it is highly recommended to add an **n** attribute holding the text title, so that it can be showed in the appropriate selector in the web edition; if `@n` is not available, `@xml:id` will be used removing the underscore characters (in other words, results may be acceptable but not always pretty …). Other elements may be used as necessary per specific needs, f.i. `@type` to record if the text is in prose or verse, but they aren't relevant for EVT. Example:

  ```
  <text n="Dream of the Rood" type="verse" xml:id="DOTR">
  ```

  Note that adding `subtype="edition_text"` is not necessary anymore, since all `<text>`s are now automatically sorted out, be they single texts or part of a larger `<group>`.

Important: in EVT 1.0 the most usual container where to put the necessary metadata described above was a simple <div> holding the actual body of a given text (e.g. `<div n="Dream of the Rood" subtype="edition_text" type="verse" xml:id="DOTR">`). This isn't possible anymore in EVT 1.1, but as you can see for existing documents the solution is simple: just copy the relevant attributes in the corresponding <text> element.

- Starting from version 1.1 EVT supports the case when there is some prefatory or introductory matter before the main body of text in the original document, for which digitized images are possibly available to be showed together with the edition text. Since all prefatory matter, both present in the original document and/or added by the editor, is encoded in the **`<front>`**, it is therefore necessary to distinguish which is which:

  - prefatory matter already present in the original document should go in a <div> having a type="document_front"attribute; the first `<pb/>` must be the first node inside this <div>;
  - all other <div>s are going to be treated as text added by a modern editor, and will be inserted in the "Info" or "Regesto" text frame.

The text-image linking is done exactly in the same way as for the main text as described above, but there is a caveat: if the prefatory text ends exactly in the same page where the main text begins, it is necessary to add two `<pb/>`s:

- one at the end of the `<front>` with an `@xml:id` ending with -front;
- one at the beginning of the `<body>` with the same `@xml:id` but without the -front suffix.

Example:

```
<front>

  <pb n="1" xml:id="G_vol1_001-front"/>

  <docTitle>
    <titlePart type="main">ARLEQUIN<lb/> MERCURE GALANT.</titlePart>
    <titlePart type="sub">COMEDIE EN TROIS ACTES.</titlePart>
  </docTitle>
  <performance>
  <p rend="italic">Mise au Theâtre par Monsieur D*** et representée pour
```

```
    la premiere fois par les Comediens Italiens du Roy dans leur Hostel de
    Bourgogne, le 22. Janvier 1682.</p>
  </performance>
</front>
<body>

  <pb n="1" xml:id="G_vol1_001"/>

  <head>SCENES FRANÇOISES<lb/>D'ARLEQUIN<lb/>MERCURE GALANT.</head>

  <div type="scene" n="1" xml:id="G_I_01_s1">
  <head type="scene">SCENE DES NOUVELLES.</head>
```

Important: at the present moment this feature only works for unitary TEI documents, i.e. documents having a single <text> and **not** the <text> <group> <text>s hierarchy. Also note that this only applies to the single <pb/> present both in the <front> and in the <body> of a text, all other <pb/>s in the <front> do **not** need a -front suffix.

- **`<pb/>`** elements are used to mark up folio sides: they must include the `@n` and `@xml:id` attributes, respectively to show the correct folio number and to enable text-image linking at the page level (see above *1.1 Single Side Images at standard resolution*); more in detail, the **`@n`** is the label you will see in the page selector in the web interface, so you are relatively free to choose the characters that you put in it, while the **`@xml:id`** is a unique identifier used to recover the text and image of each page (you can't have two identical values for `@xml:id` in your file, while you can have two or more identical `@n` attributes). For the Bookreader view to work properly, the `@n` attribute values must end with "r" (for "recto") o "v" (for "verso").

  ```
  <pb n="104v" xml:id="VB_fol_104v"/>
  ```

Important: if a document starts on a new page/folio, then the corresponding `<pb/>` must be the first node inside the `<body>` or the first node of the first `<div>` inside the `<body>`. In other words, always start a text with the relevant `<pb/>`.

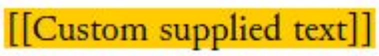
- **`<lb/>`** elements are used to mark up manuscript lines: they must include the `@n` and `@xml:id` attributes, respectively for line numbering and text-image linking at the line level; `@xml-id` only serves the purpose of text-image linking so you can omit it if you're not going to take advantage of this feature.

```
<lb facs="#VB_line_104v_07" n="7" xml:id="VB_lb_104v_07"/>
```

- Starting from version 1.2 EVT supports use of the **`<supplied>`** element according to two different `@reason` values: `<supplied reason="omitted">` (= text which has been omitted by the scribe, usually one or more characters, will be inserted in angle brackets < >) and `<supplied reason="illegible">` (= text which is currently illegible because of damages to the manuscript or textual transmission problems, will be inserted in square brackets [ ]). If no `@reason` is specified, the default is the second value ("illegible") and text supplied by the editor will be inserted into square brackets.
  It is possible to customize the appearance of `<supplied>` elements for other `@reason` values: you just have to insert your own CSS rules in the `config/evt_builder-custom-styles.css` file (see below) similar to the following:

```
/* layout for the element <supplied reason="custom-reason">  */
span.interp-supplied[data-reason='custom-reason'] {
   background: #0000;
}
/* content before the element <supplied reason="custom-reason">  */
span.interp-supplied[data-reason='custom-reason']::before {
   content: "[[";
}
/* content after the element <supplied reason="custom-reason">  */
span.interp-supplied[data-reason='custom-reason']::after {
   content: "]]";
}
```

  Following these rules, all elements such as, for instance, `<supplied reason="custom-reason">Custom supplied text</supplied>` will be formatted in this way:


[[Custom supplied text]]

- **`<ref>`** elements with a `@target` including the “www” or “http” strings will be transformed into HTML hyperlinks.

2.3 Embedded Transcription

The Embedded Transcription method, for which EVT provides experimental support thanks to an EADH small grant, is an alternative way to attach a transcription to an image, be it on parchment or other types of support. This is what EVT expects to find in a TEI XML document created according to this method:

- the TEI document must be based on one or more `<sourceDoc>` elements, with no `<facsimile>` or `<text>`;

- cartesian coordinates for text-image link at line level are once again placed in the relevant attributes of `<zone>` elements (see the preceding section);
- folio sides marked with the `<surface>` element that are direct children of `<sourceDoc>` or `<surfaceGrp>` must include the `@xml:id` attribute. If you want to activate the Bookreader view, the `@xml:id` attribute values end with "r" (for "recto") or "v" (for "verso");
- transcription text goes into `<line>` elements within or at the same hierarchy level of `<zone>` elements: we recommend to put them inside the corresponding `<zone>`s, alternatively they may be linked to `<zone>`s by means of their `@xml-id`; empty `<line>`s will be ignored.

Full example:

```
<sourceDoc xml:id="DOTR_ET">
  <surface xml:id="VB_fol_104v" n="104v">
    <graphic height="1800px" url="immagini\Vercelli-Book_104V_S_300dpi.jpg"
          width="1200px"/>
    <zone lrx="1108" lry="560" rend="visible" rendition="Line" ulx="210"
          uly="459">
      <line n="7" xml:id="VB_txtline_104v_07"><hi rend="init3.1">
          <g ref="#Hunc"/></hi><hi rend="cap">W</hi>æt ic <g
          ref="#slong"/>wefna c<g ref="#ydot"/><g ref="#sins"/>t secgan
          wylle
        <choice>
          <sic>hæt</sic>
          <corr resp="Grein">hwæt</corr>
        </choice>
        <choice>
          <orig>mege mætte</orig>
          <reg>me gemætte</reg>
        </choice></line>
      <line n="8" xml:id="VB_txtline_104v_08">to midre nihte syðþan <choice>
        <orig>reord b<g ref="#eenl"/>r<g ref="#eenl"/>nd</orig>
        <reg>reordberend</reg>
      </choice> reste wunedon<orig><pc type="metrical">.</pc></orig></line>
    </zone>
    <zone corresp="#VB_txtline_104v_09" lrx="1031" lry="610" rend="visible"
          rendition="Line" ulx="257" uly="558" xml:id="VB_msline_104v_09"/>
    <line facs="#VB_msline_104v_09" n="9" xml:id="VB_txtline_104v_09">þuhte me
          þæt ic <choice>
          <orig>ge <g ref="#sins"/>awe</orig>
          <reg>gesawe</reg>
        </choice>
        <g ref="#sins"/>yllicre treow<choice>
          <orig>onlyft</orig>
          <reg>on lyft</reg>
        </choice></line>
    […]

    <zone corresp="#VB_txtline_105r_32" lrx="987" lry="1476" rend="visible"
          rendition="Line" ulx="121" uly="1422" xml:id="VB_msline_105r_32"/>
    <line facs="#VB_msline_105r_32" n="32"
          xml:id="VB_txtline_105r_32"><choice>
          <orig>ge namon</orig>
          <reg>genamon</reg>
        </choice>
        <damage type="faded">hie</damage> þær ælmihtigne <choice>
            <orig>god</orig>
            <reg>God</reg>
          </choice> <g ref="#aacute"/>hofon hine of ðam</line>
  </surface>

</sourceDoc>
```

2.4 Support for named entities

Starting from version 0.2.0 EVT supports named entities. To best take advantage of this feature it is necessary to prepare lists that will be included in the `<teiHeader>`. One likely candidate is a list of persons, as in this example (selected from the Codice Pelavicino digital edition):

```
<listPerson>
  <person xml:id="AccursetusVitalis">
    <persName>
      <forename>Accursetus</forename>
      <surname>quondam Vitalis de castro Sarzane</surname>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="AcoltusBonavite">
    <persName>
      <forename>Acoltus</forename>
      <surname>quondam Bonavite de Trebiano</surname>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="Adiutus">
    <persName>
      <forename>Adiutus/Aiutus</forename>
    </persName>
    <sex>M</sex>
  </person>
  <person xml:id="Adiutuscap">
    <persName>
      <forename>Adiutus / Aiutus</forename>
    </persName>
```

```
      <sex>M</sex>
      <occupation n="2">presbiter, capellanus domini Guilielmi</occupation>
    </person>
    <person xml:id="Adiutusnot">
      <persName>
        <forename>Adiutus / Aiutus</forename>
      </persName>
      <sex>M</sex>
      <occupation n="1">sacri palatii notarius</occupation>
    </person>
    <person xml:id="Adornellusnot">
      <persName>
        <forename>Adornellus</forename>
        <surname>domini Tranchedi comitis de Advocatis de Luca</surname>
      </persName>
      <sex>M</sex>
      <occupation n="2">iudex ordinarius, notarius</occupation>
    </person>
  [...]
  </listPerson>
```

Note that every `<person>` item relates to an individual, in this case historical figures, but they might as well be fictional characters. Another useful list may one of places (again from the CP edition):

```
  <listPlace>
    <place xml:id="Aciliano">
      <settlement type="località">Aciliano</settlement>
    </place>
    <place xml:id="Acola">
      <settlement type="località">Acola</settlement>
    </place>
    <place xml:id="Agina">
      <settlement type="località">Agina</settlement>
    </place>
    <place xml:id="Alione">
      <settlement type="località">Alione</settlement>
    </place>
    <place xml:id="Alpes">
      <settlement type="monte">Alpes</settlement>
      <placeName type="new">Alpi Apuane</placeName>
    </place>
  [...]
  <listPlace>
```

At the present moment, EVT supports the following list elements for the purpose of named entities annotation and linking:

- `<listPerson>`
- `<listPlace>`
- `<listOrg>`

Note that the TEI schemas offer other specific list elements, such as `<listNym>` for the canonical form of names, or `<listEvent>` for historical events, but it is also possible to use the generic `<list>` element to provide for other needs (you only have to specify a value for @type to make explicit what kind of items are gathered in a list), provided that all lists are related to individual "objects". Future versions of EVT will support other types of list, including the generic one.

Once the lists are ready, you can link each occurrence of an item to the corresponding list entry:

```
<p xml:id="CCLXXXXII_p_003" n="3"><ptr target="CCLXXXXII_st_001"
facs="#st_279r_001"/> Ego <persName ref="#Adiutusnot">Adiutus,
<roleName>sacri palacii notarius</roleName></persName>, hiis interfui et
de mandato suprascripti domini episcopi hanc cartam abreviavi et in publicam
formam redegi, signum et nomen proprium apponendo</p>
```

2.5 Notes

2.5.1 Stand-off notes

Starting from version 1.2, it is possible to add notes not in the usual inline fashion, but in a separate part of the TEI document, typically grouping them in the `<back>` element; this is particularly useful when the same <note> may refer to more than one text segment in the document, to avoid redundancy.

To take advantage of this feature, you will have to add an **`@xml:id`** to each `<note>` element, and insert a `<ptr/>` with a **`@type="noteAnchor"`** attribute and a **`@target`** attribute whose value will be the corresponding `<note>`'s ID (as usual preceded by the "**#**" character).

Example:

```
<lb/>Lucchese de <placeName>sancto<lb/>Luca</placeName></persName>.
<ptr target="#SM_note_1" type="noteAnchor"/>

...

<note n="1" xml:id="SM_note_1">Per la chiesa di San Luca, nell'area
extramuraria orientale, esterna alla <foreign xml:lang="lat">civitas</foreign>
altomedievale, si veda <ref target="#Garzella1991">Garzella 1991, pp. 179,
188</ref> e <emph>passim</emph>.</note>
```

2.5.2 Critical notes

Starting from version 1.1 EVT lets the editor distinguish between critical notes and commentary notes: the former are encoded as `<note type="critical">`, the latter as `<note type="comment">`; if the type attribute is missing, the note will be considered a comment.

2.6 Tables and columns

Starting from version 1.2, EVT supports the use of the **`<table>`** element, possibly with a layout of two or more columns. A table doesn't require special precautions compared to the normal TEI usage. The use of **`<cb/>`** to mark columns, on the other hand, needs more attention: you will have to add both an `@xml:id` and a `@rend` attribute to each `<cb/>` marking the beginning of a column; the latter must have a value corresponding to the actual column layout in the document, e.g. "**2col_layout**" for two columns, "**3col_layout**" for three, etc. This feature is currently being worked upon, see the "Known bugs" section for a caveat.

Full example:

```
<p xml:id="CCCXLVI_p_002" n="2">Sunt ulterius episcopi:
  <pb n="305v" xml:id="fol_305v"/>
  <cb n="1" rend="2col_layout" xml:id="CCCXLVI_cb_003"/>
  <table rows="1" cols="2">
    <head>§ In <placeName ref="#Graçano">Graçano</placeName></head>
    <row role="data">
    <cell><persName ref="#Ventura">Ventura</persName></cell>
    <cell><measure type="Imperiales" quantity="1" unit="denari">denarium
    I<note xml:id="CCCXLVI_n_006" n="f" place="foot"><term
    xml:lang="latin">Ventura denarium I</term>: <emph>nel testo</emph>
    <term xml:lang="latin">denarium I Ventura</term>.</note></measure></cell>
    </row>
…
  <cb n="2" rend="2col_layout" xml:id="CCCXLVI_cb_004"/>
    <row role="data">
    <cell><persName ref="#PerasonusAventia">Perasonus de <placeName
    ref="#Avenza">Aventia</placeName></persName></cell>
    <cell><measure type="Imperiales" quantity="4" unit="denari">denarii
    IIII</measure></cell>
    </row>
    ...
    <row role="data">
    <cell>Filie <persName ref="#Lanfranchinus">Lanfranchini</persName></cell>
    <cell><measure type="Imperiales" quantity="1" unit="denari">denarium
    I</measure></cell>
    </row>
  </table>
  <cb/>
</p>
```

2.7 Edition levels

Different **edition levels** in the same TEI document are managed through a combination of transcriptional and editorial elements:

- the **diplomatic** level is encoded using the `<damage>`, `<hi>`, `<abbr>`+`<am>` and `<orig>` elements; at the character level, if a `<charDecl>` is present, using the `<mapping type="diplomatic">` character values inside each <char> (or <glyph>) element;
- the **interpretative** level is encoded using the `<supplied>`, `<expan>`+`<ex>` and `<reg>` elements; at the character level, if a `<charDecl>` is present, using the `<mapping type="normalized">` character values inside each `<char>` (or `<glyph>`) element; `<sic>`/`<corr>` pairs are possible, but not necessary if a full critical edition is envisaged.

Note that all the editorial elements are usually inserted in <choice> elements: this is mandatory for word-level pairs.

Full example:

```
<text>
 <body>
   <div n="DOTR" subtype="edition_text" type="verse" xml:id="DOTR">
    <pb n="104v" xml:id="VB_fol_104v"/>
     <l n="1"><lb facs="#VB_line_104v_07" n="7" xml:id="VB_lb_104v_07"/>
           <hi rend="init3.1"><g ref="#Hunc"/></hi>
           <hi rend="cap">W</hi>æt ic
           <g ref="#slong"/>wefna c<g ref="#ydot"/>
           <g ref="#sins"/>t secgan wylle</l>
     <l n="2"><choice>
           <sic>hæt</sic>
                 <corr resp="Grein">hwæt</corr>
              </choice>
               <choice>
                 <orig>mege mætte</orig>
                 <reg>me gemætte</reg>
               </choice>
           <lb facs="#VB_line_104v_08" n="8" xml:id="VB_lb_104v_08"/>
           to midre nihte</l>
            <l n="3">syðþan <choice>
                 <orig>reord b<g ref="#eenl"/>r<g ref="#eenl"/>nd</orig>
                 <reg>reordberend</reg>
               </choice> reste wunedon<orig><pc type="metrical">.</pc></orig></l>
             <l n="4"><lb facs="#VB_line_104v_09" n="9"
     xml:id="VB_lb_104v_09"/>þuhte me þæt ic <choice>
                 <orig>ge <g ref="#sins"/>awe</orig>
                 <reg>gesawe</reg>
```

```
                </choice>
                <g ref="#sins"/>yllicre treow</l>

        […]

            <l n="156">ælmihtig <name type="religion"><choice>
                <orig>god</orig>
                <reg>God</reg>
              </choice></name> þær hi<g ref="#sins"/> eðel wæ<g ref="#sins"/>
            <g ref="#colmidcomposit"/></l>
        </div>
      </body>
    </text>
```

3. Configuring EVT

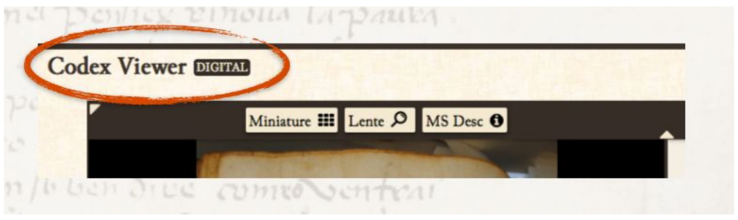
3.1 The Configuration File

If you open the `evt_builder-conf.xsl` file that is inside the `config` folder, you will be able to configure the existing parameters to customize the layout and functionalities of the resulting web-edition. The parameters that you will be able to configure are the following:

Edition/Project title and Badge

```
<!-- EN: Index title -->
<xsl:param name="index_title" select="'Codex Viewer'"/>
<xsl:param name="badge_text" select="'DIGITAL'"/>

<!-- EN: Hide/Show badge -->
<xsl:param name="badge" select="true()"/>
```



Web site of the project

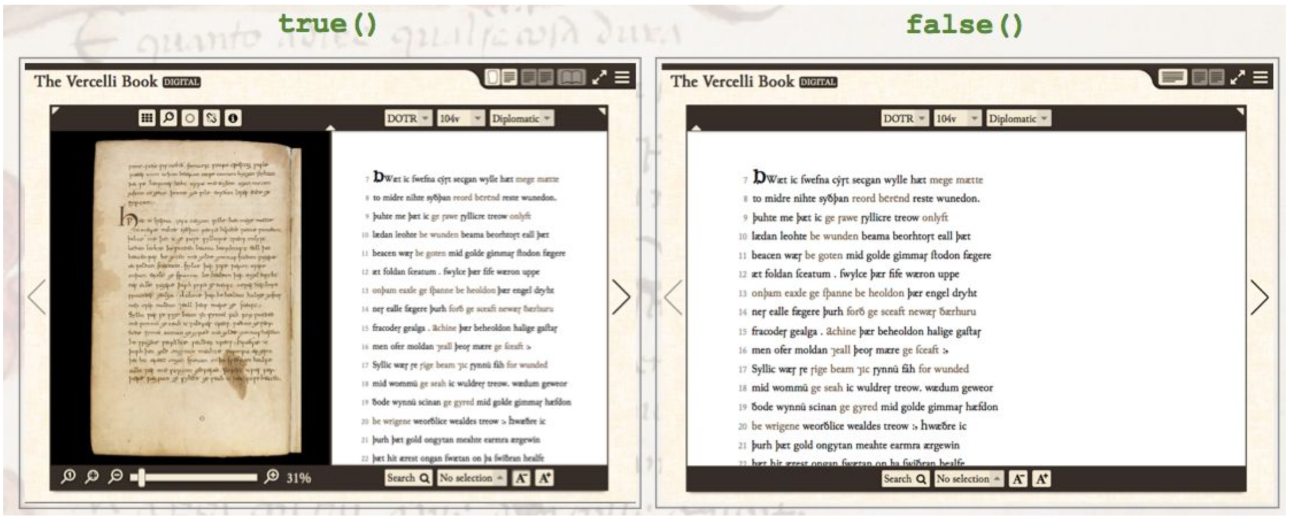
```
<xsl:param name="webSite" select="'http://www.yoursite.com/'" />
```

Image extension

```
<xsl:param name="imageExt"/>jpg</xsl:param>
```

Image frame (on/off)

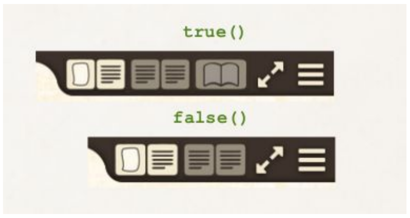
```
<xsl:param name="image_frame" select="true()"/>
<xsl:param name="image_frame" select="false()"/>
```



ÐWæt ic ſwefna cẏꞃt secgan wylle hæt mege mætte
to midre nihte syðþan reord berend reste wunedon.
þuhte me þæt ic ge ꞃawe ꞃyllicre treow onlyft
lædan leohte be wunden beama beorhtoꞃt eall þæt
beacen wæꞃ be goten mid golde gimmaꞃ ſtodon fægere
æt foldan ſceatum . ſwylce þær fiſe wæron uppe
onþam eaxle ge ſþanne be heoldon þær engel dryht
neꞃ ealle fægere þurh forð ge sceaft newæꞃ ðærhuru
fracodeꞃ gealga . ac hine þær beheoldon halige gaſtaꞃ
men ofer moldan ⁊eall þeoꞃ mære ge ſceaft :-
Syllic wæꞃ ꞃe ꞃige beam ⁊ic ꞃynnū fah for wunded
mid wommū ge seah ic wuldreꞃ treow. wædum geweor
ðode wynnū scinan ge gyred mid golde gimmaꞃ hæfdon
be wrigene weorðlice wealdes treow :- hwæðre ic
þurh þæt gold ongytan meahte earmra ærgewin
þæt hit ærest ongan ſwætan on þa ſwiðran healfe

Book Reader view (on/off)

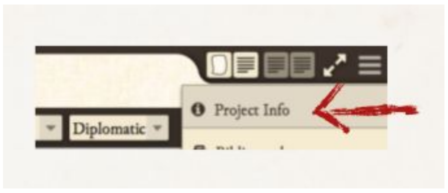
```
<xsl:param name="double_view" select="true()"/>
<xsl:param name="double_view" select="false()"/>
```



Edition levels

```
<xsl:variable name="edition_array" as="element()*">
      <edition>Diplomatic</edition>
      <!-- NB: Leave it empty if you don't have a diplomatic
      edition: <edition></edition> -->
      <edition>Interpretative</edition>
      <!-- NB: Leave it empty if you don't have
      an interpretative edition: <edition></edition> -->
</xsl:variable>
```

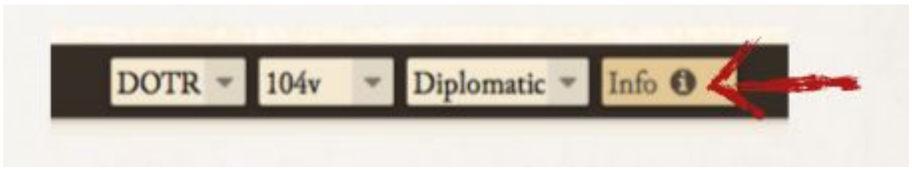
Information about the project (on/off)



```
<xsl:param name="headerInfo" select="true()"/>
```

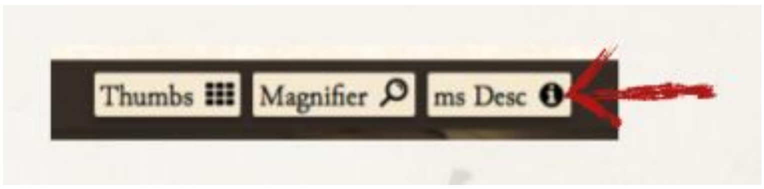
Information about the current document (on/off)

```
<xsl:param name="frontInfo" select="true()"/>
```



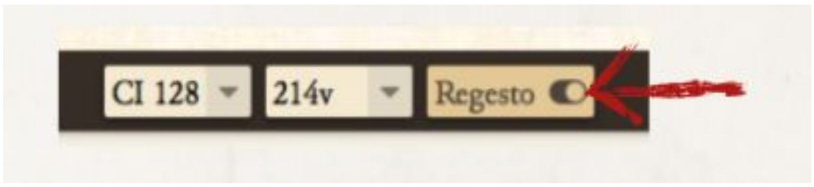
Manuscript description (on/off)

```
<xsl:param name="msDesc" select="true()"/>
```



Regesto (on/off)

```
<xsl:param name="regesto" select="true()"/>
```



Frame content showed at first loading of the edition

Left frame: possible values are

- *image*: if you want to show the digitized ms images
- *info*: if you want to show manuscript information

```
<xsl:variable name="left_frame_default_content" select="'image'" />
```

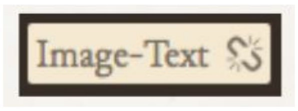
Right frame: possible values are

- *text*: if you want to show the edition text
- *info*: if you want to show information about the current text

```
    <xsl:variable name="right_frame_default_content" select="'text'" />
```

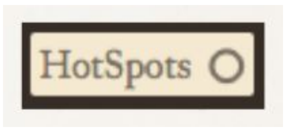
Image-text linking button (on/off)

```
<xsl:param name="txtimg_link_button" select="true()"/>
```



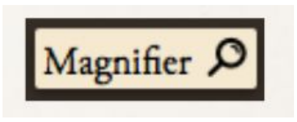
Hot-spot button (on/off)

```
    <xsl:param name="hs_button" select="true()"/>
```



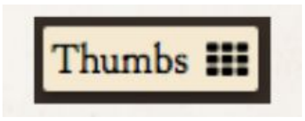
Magnifying lens button (on/off)

```
    <xsl:param name="mag_button" select="true()"/>
```



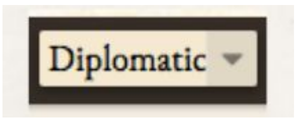
Thumbnails button (on/off)

```
    <xsl:param name="thumbs_button" select="true()"/>
```



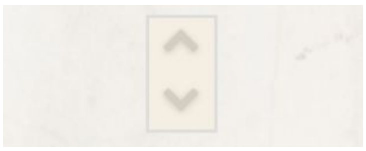
Edition level selector

```
    <xsl:param name="edition_level_selector"
               select="true()"/>
```



Document navigation button (on/off)

```
    <xsl:param name="document_navigation"
    select="true()"/>
```



Search and virtual keyboard (on/off)

```
    <xsl:param name="search" select="true()"/>
    <xsl:param name="virtual_keyboard_search" select="true()"/>
```

Page selector position (left/right)

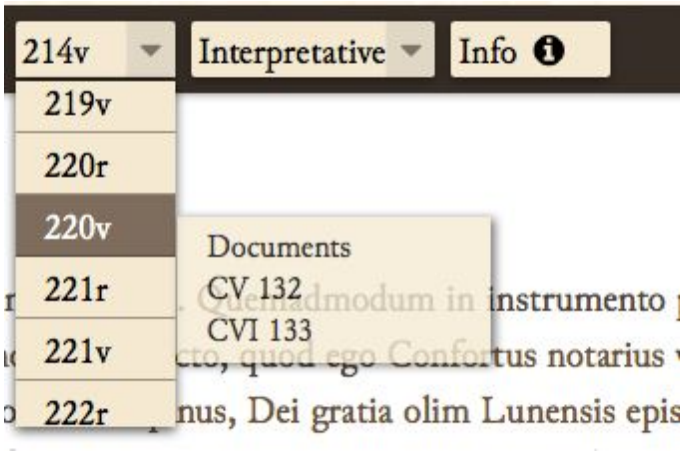
```
    <xsl:param name="pp_selector_pos" select="'right'"/>
```

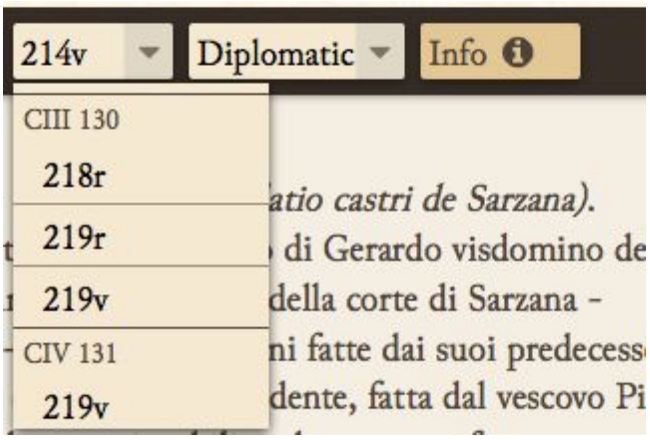
Page grouping based on document (on/off)

```
    <xsl:param name="pp_selector_doc_grouping" select="false()"/>
```

Tooltip for page selector

```
    <xsl:param name="pp_selector_doc_tooltip" select="true()"/>
```





Person list (on/off)

```
    <xsl:param name="list_person" select="true()"/>
```

Place list (on/off)

```
    <xsl:param name="list_place" select="true()"/>
```

List of organizations (on/off)

```
    <xsl:param name="list_org" select="true()"/>
```

Customizable filters

```
    <xsl:variable name="lists" as="element()*">
        <persName/>
        <placeName/>
        <roleName/>
        <measure/>
        <date/>
        <orgName/>
    </xsl:variable>
```

Note that if you don't want the "Entities selector" list to appear in the text toolbar you only have to delete all elements within the `<xsl:variable>`, maintaining the variable empty as follow:

```
    <xsl:variable name="lists" as="element()*"></xsl:variable>
```

All available options are described by means of comments in the file itself and should be understandable for anyone having a little experience with XML/XSLT documents.

3.2 Fine-tuning the results, custom templates

Bookreader view

The Bookreader navigation needs the identification of page pairs, which is based on the `@n` values of the `<pb/>`/`<surface>` elements. If you want EVT Builder to properly recognize the page pairs, you need to use numbers instead of letters for such values, followed by "v" for *verso* and "r" for *recto*. We do realize that this is a bit restrictive, but we are working on a better way to recognize the pairs. It is possible, in any case, to manually modify the `structure.xml` file (in the `output_data` folder) to correct possible mistakes: you'll find a `<pages>` element inside that file, with `<pair>`(s) containing two `<pb>`(s), one for the image in the left-hand-side, one for the image in the right-hand-side. Note that every time you transform the XML file with EVT Builder, the structure file will be overwritten and you will lose your changes.

The keyboard_config.xml file

In the `config` folder you will find a `keyboard_config.xml` file which will let you configure and use, provided that you set the corresponding switch to `true()` in the `evt_builder-conf.xsl` file, a virtual keyboard to insert special characters in the search field. Simple instructions are included in the file itself.

The language files

In the `config/langpack` folder there are a few files with a .js extension that are used to localize the User Interface language, these correspond to the languages you can choose from the Settings menu (first item on the main menu starting from right). You can change the language values to correct mistakes or to add a new language, in the latter case please send us the new .js file!

The `evt_builder-custom-templates.xsl` file

If you just need an XSLT template to handle a specific TEI element not included in the transformation chain, you can add it in this file, `evt_builder-custom-templates.xsl` in the `config` folder. When adding your templates to this file don't forget to add one of the three available modes (interp, dipl) so that the template will be included in the transformation chain to operate in the appropriate mode. If you want your template to be applied in all available modes it is necessary to add all possible values for `@mode`; it is also advised to set a very high priority for your template, so that it will be executed avoiding possible conflicts with EVT's standard templates.

Example:

```
<xsl:template match="tei:title" mode=" interp dipl" priority="9">
     ...
</xsl:template>
```

Beware of possible conflicts if you try to modify the processing of a node already included in the standard templates.

The `evt_builder-custom-styles.css` file

This configuration file has exactly the same purpose as the .xsl one described above: this is where you can add custom CSS rules for specific elements.

`evt_builder-custom-styles.css` can be found in the `config` folder. You can use it together with the custom XSLT templates file, but again be careful if trying to modify standard rules.

The structure.xml file

For immediate fixes or temporary changes to the file and directory structure of the web edition you can also modify this file (in the `output_data` folder), note however that all changes will be lost when a new version of the edition is generated!

4. Step by step instructions

How to use the EVT builder pack, step by step:

1. Put the images files of every manuscript folio in the `data/input_data/images` folder. The images must be organized as explained above:

- single side images go in the `data/input_data/images/single` folder
- double side images go in the `data/input_data/images/double` folder
- hotspot images go in the `data/input_data/images/hotspot` folder

2. Put the TEI XML file of the transcription and all other related files if necessary (e.g. schema, entities) in the `data/input_data/text` folder. You can use a sub-folder e.g. for the schema, provided that the correct path to the schema is included in the TEI XML documents.

3. Delete, or move into another folder on your hard drive, all content of the `data/output_data/` folder and the `index.html` file. Note that this is just a precaution and/or a way to backup your previous results, the transformation should overwrite the existing folder in any case.

4. Configure the software as explained in section *3. Configuring EVT.*

5. Start the edition creation process by applying the `evt_builder.xsl` stylesheet, which is in the `builder_pack` folder, to the TEI XML transcription document. To perform the XSLT transformation using Oxygen XML Editor:

- Open the XML file with Oxygen;
- Select the "Document/Transformation/Configure a transformation scenario" menu voice;
- Choose the "XML transformation with XSLT" scenario;
- Clone a basic scenario such as "TEI P5 XHTML";
- Type or select the path to the `evt_builder.xsl` file;
- Choose one of the 9.x versions of the Saxon XSLT transformer engine (for example Saxon-EE 9.1.5.1)
- Save and apply the transformation scenario.

Known bugs

No software tool is usually totally bug-free after a certain complexity threshold has been crossed, and surely EVT is no exception under this regard. All the known bugs and quirks will be detailed here, please make sure that you send us an email about any problem you may face, so that if it is a new bug we have a chance to try to reproduce it and then fix it.

If the encoded text is split up in two or more sub-`<div>`s inside a main `<div>`, it may be duplicated at the moment of the XSLT transformation. For the moment being, this bug has been fixed changing the relative rule in the `evt_builder-conf.xsl` file (inside the `config` folder) to make it more “tolerant” if sub-divs are present. A final fix, so that all risk of content duplication will be avoided, will be introduced in the next EVT version.

There may also be a problem in case of duplicated `<lb/>`s inside `<choice>` elements, leading to an error in the XSLT transformation chain. This is quite a rare bug, not occurring under 'normal' circumstances, that will be fixed in a point release for version 1.0.

Support for <cb/> elements is not yet perfect: to make it work properly, you may have to add a final, attribute-less `<cb/>` at the end of the columnar layout area of the original document (see section 2.6).

Warnings

Note that, while a web-edition created with EVT is ready to be copied on a web server for publication, you may want to prevent access to part of the directory structure: this is particularly the case of the data/input_data/images directory, where normal and high resolution images of the edited manuscript can be found.

In EVT 1.1 the starting point for the XSLT transformation chain is now each <text> element in your TEI document: this marks a difference with EVT 1.0, so please read the appropriate section (2.2 Parallel Transcription, third item in the list) especially if you have TEI documents that already work in EVT 1.0 and you want to update them so that v. 1.1 can be used.

References

EVT home page
EVT release page
EVT development repository
Mail contact

Leclerc, Élise. 'Le risorse dell'italianista 2.0: Edition Visualization Technology.' Blog post: Fonte gaia blog 2015/08/25. URL: http://fontegaia.hypotheses.org/1239.

Rosselli Del Turco, Roberto. 'EVT development: an update (and quite a bit of history).' Rapporto tecnico sul blog *Edition Visualization Technology*, 26 gennaio 2014. URL: http://visualizationtechnology.wordpress.com/2014/01/26/evt-development-an-update-and-quite-a-bit-of-history/.

Rosselli Del Turco, Roberto et al. "Edition Visualization Technology: A Simple Tool to Visualize TEI-Based Digital Editions." *Journal of the Text Encoding Initiative* Issue 8 (2015). URL: http://jtei.revues.org/1077; DOI: 10.4000/jtei.1077.

TEI Consortium, eds. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. 3.0.0. Last updated on 29th March 2016. TEI Consortium. http://www.tei-c.org/Guidelines/P5/.