Word to TEI

This is a sample document to test conversion of word docx to TEI using the TEI processing model. It was generated by uploading the word document located in data/doc/test.docx in your local TEI Publisher installation. You can also download it from the [TEI Publisher website](https://teipublisher.com/exist/apps/tei-publisher/data/doc/test.docx), edit it and upload it via the upload panel on the start page of your local TEI Publisher.

# Style Conversion

Rather than trying to convert everything, the default ODD transformation attempts to preserve the semantics of the text. Most style properties are thus ignored. This is by intention: trying to preserve as much as possible would likely just add noise and result in low-quality TEI.

Users are free to extend the default ODD with additional heuristics. For example, a paragraph being entirely bold could also be treated as a heading, or a left text indent may indicate a quote. The default ODD being used for Word to TEI transformations is called docx.odd.

Empty paragraphs like the one below will be removed from the docx before it is being passed to the ODD for transformation.

## Inline Styles

Text styled as italic, underline or bold will be transformed into a tei:hi with a corresponding @rend attribute. Other types of formatting will be ignored.

Inline styles whose name starts with „tei:“ are transformed into TEI elements with the same name. So if a character sequence uses a style called „tei:persName”, it will be wrapped into a TEI persName element in the output, e.g. Johann Wolfgang Goethe. A place name can be marked with a style „tei:placeName” and should be transformed accordingly: Frankfurt, Berlin, München. And damaged text could be encoded by applying a style „tei:supplied“.

There’s also a default convention for encoding additional attributes: text in angle brackets will be interpreted as a list of attribute=value pairs. Multiple items should be separated with a “;”. For example, to set a @rend and provide a @ref for a placeName, you can write Frankfurt<rend=smallcaps;ref=Frankfurt am Main>[[1]](#footnote-1).

This notation requires quite some typing. You may always extend the ODD with additional rules for easier conventions though. For example, if persName does always need a @ref attribute in your edition, you could have a simplified rule which parses: Friedrich Dürrenmatt<118527908>.

Because Word has a tendency to split character ranges at random points, some pre-processing is applied before the docx is passed to the ODD for processing: subsequent ranges referencing the same character style are combined by nesting them into an additional w:r range element, which references the common character style and the style is then removed from the individual ranges.

You can thus safely assume within the ODD that the content of a range includes all sibling text using the same character style.

By design, Word does not support nested character styles. It is thus not possible to e.g. mark up a persName inside a supplied. The standard character styles for **bold**, *italics* and underline are preserved though – like in the following paragraph which is marked up as supplied:

This is supplied text containing **bold**, *italics* and underline. We may also encounter local formatting inside a persName like Friedrich ***Dürrenmatt***<118527908>.

## Paragraph Styles

### Headings

Word does not have a concept for text division, so we have to reconstruct them:

1. Paragraph styles starting with „heading“, „title“ or „subtitle“ generate a tei:head. The outline level assigned to the heading is recorded as well.
2. In a second pass through the generated output, divs are generated based on the outline level:
   1. A div spans all text from the heading to the next heading on the same outline level
   2. Repeat the process for all headings within this division on a lower outline level

### Footnotes

Footnotes are translated into TEI note elements[[2]](#footnote-2). We also support endnotes[[3]](#endnote-1), which are transformed into a note with type=“endnote“.

### Lists

Lists are tricky, because Word essentially just stores list items in a flat list. Reconstructing nesting thus requires looking at the list level associated with every item. Simple lists are easy:

* A list item[[4]](#footnote-3)
* Another list item

We can also have numbered lists, which are translated into a <list type=“ordered“>:

1. First item in a numbered list
2. Second item in a numbered list

Nested lists are quite tricky:

* A bullet list item

1. A nested list item
2. Another nested list item
   1. Nested in a nested list
   2. More …

* A *second* bullet list item

Paragraph styles starting with „tei:“ are converted to TEI elements with the same name. For example, the lines of the following poems are assigned the style “tei:l”:

I love to heare her speake, yet well I know,

That Musicke hath a farre more pleasing sound:

I graunt I never saw a goddesse goe,

My Mistres when shee walkes treads on the ground.

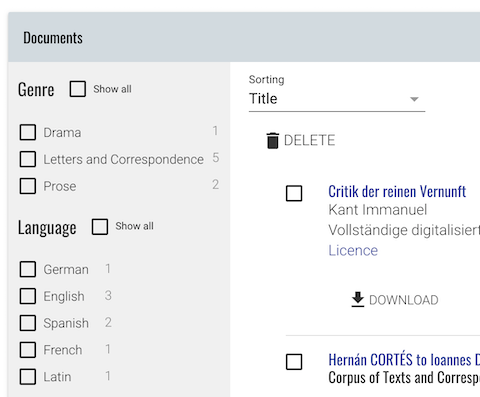
# Tables

We can do simple tables very well. Spanning multiple colums is also easy, but things become more difficult for row spans, which are not implemented yet.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Hours | Hourly rate | Price |
| Customize ODD | 3 | 120 | 360 |
| Generate App | 4 | 120 | 480 |
| Test and Deploy | 2 | 120 | 240 |
| Total | | | 1080 |

# Embedded Images

Below image will be embedded:



Inside eXist, images are copied into a subcollection starting with the name of the docx file being processed and suffixed with *.media*.

1. Text content in angle brackets will be automatically stripped from an inline element by the post-processing step, so you do not need to handle this within the ODD. [↑](#footnote-ref-1)
2. A *footnote* may contain inline **formatting**. [↑](#footnote-ref-2)
3. This is an *endnote*, which should appear at the very end of the text. [↑](#endnote-ref-1)
4. And here we have a footnote with a [link](#target1) to another place in the document. [↑](#footnote-ref-3)