Quick TEI (QTEI) - a lightweight tool for TEI documents

Schepp, Moritz
schepp@wendig.io
Wendig OÜ, Estonia

Wübbena, Thorsten
wuebbena@ieg-mainz.de
Leibniz Institute of European History, Germany

Starting situation

Over the past decades, TEI/XML has established itself as a document format for the encoding and exchange of texts in the processing and provision of textual historical sources. Various tools are available for encoding data in TEI – from simple to specialised XML editors, from open source to proprietary. With these tools, domain experts are able to carry out their scientific mark-up of the texts and enrich the documents. However, creating online representations from TEI/XML data usually requires technical knowledge in web development. This often makes the TEI workflow from TEI document to online publication “fragile”. When creating web views, there is currently a lack of simple, resource-saving but still expandable solutions to adequately counter this tendency.

Status quo

One possible answer to this challenge is TEI Boilerplate (https://dcl.ils.indiana.edu/teibp/). The TEI boilerplate provides immediate publishing of the material, but requires XSLT knowledge for customizations and refers to JavaScript for adding interactive functionality, which are not part of the framework. CETEIcean (https://github.com/TEIC/CETEIcean) delivers impressive TEI parsing and rendering capabilities. As a JavaScript library, it is by definition a developer tool. eXist-db is also a well-established software for publishing and working with XML content and there are modules like TEI Publisher (https://teipublisher.com/index.html) that quickly render TEI content on the platform. Setup and hosting of eXist-db requires experience with relevant server technologies. Also, in order to extend or change the representation of TEI/XML content, a first step is often to restructure the documents using XQuery. The nature of the language quickly leads to complex constructs that generate increased maintenance effort.

Idea and implementation

In response to the need for a low-threshold solution to the requirements outlined above, we designed and developed QTEI. To keep it relevant and in line with typical requirements, we will develop it as part of a representative research project. QTEI is based on two core components: web browser and JavaScript. The browser, as one of the most comprehensive software packages for handling XML, is widely used and easy to maintain by end users. JavaScript is often part of the solution for rendering TEI/XML anyway, and so with QTEI we propose a framework that can fully implement a interactive TEI web presence with this technology.

Our intent is to enable operating the product in a maintenance-free way, which is why we chose to build something that would output a static page. Based on this approach, we developed a prototype as a NPM package that can be used either as an ES6 module (https://en.wikipedia.org/wiki/ECMAScript#6th_Edition_%E2%80%93_ECMAScript_2015) in modern JavaScript or with a simple <script>-Tag through the use of Content Delivery Network (CDN). A development environment is integrated into QTEI, which allows to customize and extend the module, but also enables the domain expert to track changes to the TEI/XML directly in a web environment. The release as a NPM package also ensures that the library is available via the CDN unpkg.com so it can be incorporated into any project immediately. No special frontend build process is required and it is not necessary to learn a new frontend framework. The source code is licensed under the AGPL (https://www.gnu.org/licenses/agpl-3.0.html) and available in a GitHub repository, where the user can also find a short demo video and the documentation (https://github.com/ieg-dhr/QTEI).

Future Work

QTEI renders a TEI document within a web page without requiring developer knowledge. However, should basic JavaScript skills be available, then the functionaly can easily be extended by the means of plain old JavaScript functions (https://en.wikipedia.org/wiki/Plain_old_Java_object). Each of the following features could likely be implemented with less than 200 lines of code:

# Editor functionality either for the TEI/XML itself or in a WYSIWYG for the HTML-rendered representation
# Full-Text-Search with term matching
# Search for tagged entities (e.g. people, places)
# Extending search even further with facets and other static results like time histograms

Discussion

The applications used in the digital humanities move between two poles: There are tools developed for specific applications and those for general purposes. This “tools paradox” (Pape et al. 2012), which involves the tension between specificity and universality and thus limitations and usability versus complexity, is also present here. We are aware of the limitations of QTEI, but understand it as a lightweight tool that should serve as an addition to a greater “DH toolbox” for an ever more diverse community. The poster contribution opens up first approaches for further discussion with the prototypical software version.

Bibliography

Pape, Sebastian / Schöch, Christof / Wegner, Lutz (2012): “TEICHI and the Tools Paradox”, in: Journal of the Text Encoding Initiative [Online], Issue 2, February 2012, DOI: 10.4000/jtei.432.