TEI BY EXAMPLE



MODULE 8: CUSTOMISING TEI, ODD, ROMA

Ron Van den Branden

Edward Vanhoutte

Melissa Terras

Centre for Scholarly Editing and Document Studies (CTB) , Royal Academy of Dutch Language and Literature, Belgium, Gent, 9 July 2010

Last updated September 2020

Licensed under a Creative Commons Attribution ShareAlike 3.0 License

TABLE OF CONTENTS

2. Some Terminology.....1

2. Some Terminology

As we have seen in Module 0: Introduction to Text Encoding and the TEI, section 5.2, all TEI elements and their attributes are organised thematically in 21 higher-level modules. If we compare the TEI specification to a library, modules are the library shelves holding the elements and attributes. Each of these modules is documented in full in a dedicated chapter of the TEI Guidelines; for an overview of which chapters correspond to which modules, see section 1.1 TEI Modules of the TEI Guidelines. Most TEI modules define attributes and elements. During time, the TEI specification has grown into an ecosystem of hundreds of interrelated elements and attributes. In order to make for easier organisation and maintenance, they have been grouped into *classes*. Each class is a group of elements and attributes. Therefore, the TEI defines two kinds of classes:

Attribute classes

A group of attributes that can occur in the same place in a TEI document. The names of attribute classes all start with `att.`, followed by a name that gives an indication of the group of attributes it contains. For example, the global attributes `@rend`, `@xml:id`, and `@n` (among others) are all defined in the attribute class att.global, which name indicates that it defines global attributes, available on all TEI elements.

Model classes

A group of elements that can occur in the same place in a TEI document. The names of model classes all start with `model.`, followed by a name that gives an indication of the group of elements it contains. For example, <p> and <ab> are being grouped in the model class model.pLike, which holds all paragraph-like elements.

Classes can refer to other classes, so it becomes possible to develop a fine-grained hierarchy of element and attribute groups that all share some common features, but add their own specific features, depending on the sub-classes they belong to.

In the organisation of TEI modules, the **`tei`** module is a bit special, in that it does no more than providing the definitions of the attribute and model classes that are being used in the definitions of elements and attributes in other TEI modules. Besides these classes, the **`tei`** module defines some other low-level constructs that are being used in the definition of actual elements and attributes:

Macros

A kind of "shortcut," combining a number of frequently co-occurring model classes into a logical group of elements that can occur in the same hierarchical contexts in a TEI document. The names of macros all start with `macro.`, followed by a name that gives an indication of the logical organisation of the elements it groups. For example, the macro macro.paraContent groups elements that can occur inside paragraphs and similar elements.

Datatypes

A set of rules for the values of attributes. The names of datatypes all start with `teidata.`, followed by a name that gives an indication of the datatype. For example, the datatype teidata.count specifies a rule for attribute values that must be positive integers. When an attribute definition refers to this datatype, its value must be a positive integer.

This system of building blocks can then be used to define the core of what the TEI is all about: elements and attributes for marking up information in or about texts. For example: the TEI element <name> defines itself by:

- declaring itself as a member of the *TEI module* **`core`**. This means that when this module is included in a TEI customisation, the <name> element will be available by default.
- declaring itself as a member of the *attribute classes* att.global, att.personal, att.datable, att.editLike, and att.typed. This means that <name> will receive the attributes defined in all those classes (at least, those that are included in the TEI customisation).
- declaring itself as a member of the *model class* model.personPart. This means that it will be able to occur inside all TEI elements that define the members of this model.personPart as its contents.
- declaring its contents as the elements grouped in the *macro* macro.phraseSeq. This means that all elements included in this macro can occur inside <name>.

How these element and attribute are defined, is explained in this TEI by Example tutorial. It will cover some of the very specific elements that make up the **`tagdocs`** module that is documented in section 22: Documentation Elements of the TEI Guidelines. Yet, instead of a purely theoretical explanation of this rather technical part of the TEI Guidelines, we will illustrate this in a hands-on fashion, building a mini TEI customisation for a small encoding project, with the help of the Roma tool. Let's dive in!