Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

DARIAH-DE
Digital Research Infrastructure
for the Arts and Humanities

Referat 09:
TEI anpassen mit Roma

DARIAH-DE Tutorial *Digitale Textkodierung mit TEI*

Redaktion: Christof Schöch (Univ. Würzburg)

Version 1.0, 02/2014

Grundlage der Folien: DH@Oxford 2012

1. Warum und wie ein Schema definieren?

Wann, warum, wie TEI anpassen?

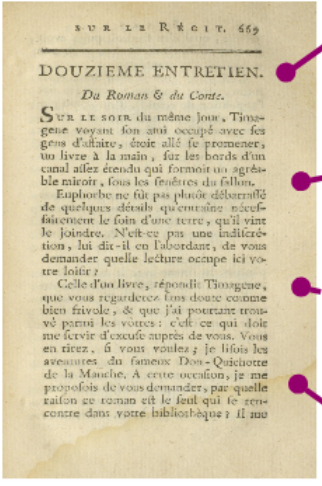
Zunächst: man passt TEI immer irgendwie an

- TEI ist umfangreich, flexibel, modular
- Konsistenz der Kodierung erhöhen
- Dokument-Modell formal ausdrücken und dokumentieren

Daher behandeln wir hier:

- Wie ist TEI aufgebaut?
- Wie definiert man ein TEI Schema?
- Wie generiert man die eigene Dokumentation?

Zu Erinnerung: Prozessmodell


SUR LE RÉCIT. 669

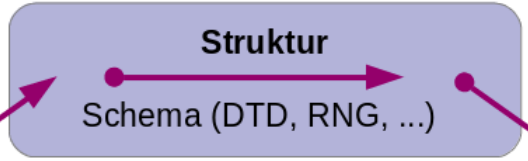
DOUZIEME ENTRETIEN.

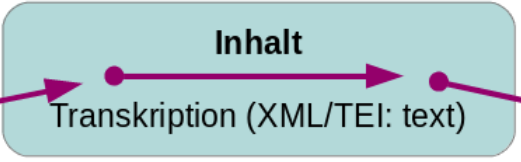
Du Roman & du Conte.

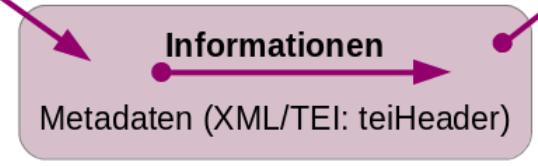
SUR LE SOIR du même jour, Timagene voyant son ami occupé avec ses gens d'affaire, étoit allé se promener, un livre à la main, sur les bords d'un canal assez étendu qui formoit un agréable miroir, sous les fenêtres du sallon.

Euphorbe ne fût pas plutôt débarrassé de quelques détails qu'entraîne nécessairement le soin d'une terre, qu'il vint le joindre. N'est-ce pas une indiscrétion, lui dit-il en l'abordant, de vous demander quelle lecture occupe ici votre loisir ?

Celle d'un livre, répondit Timagene, que vous regarderez sans doute comme bien frivole, & que j'ai pourtant trouvé parmi les vôtres : c'est ce qui doit me servir d'excuse auprès de vous. Vous en riez, & vous voulez ; je lisois les aventures du fameux Don-Quichotte de la Manche. A cette occasion, je me proposois de vous demander, par quelle raison ce roman est le seul qui le rencontre dans votre bibliothèque ? Il me







Etwas Terminologie

Allgemein

- TEI besteht aus Modulen; siehe Input „Guidelines 1-23”
- Jedes Modul enthält „element specifications”

Jede Element-Spezifikation enthält:

- einen Element-Namen („canonical name” / „generic identifier”; <gi>);
- eine kanonische Beschreibung
- Angabe zur Klassenzugehörigkeit
- Angabe zum Kontext
- Definition der Attribute
- Definition des „content model”
- Beispiele und Anmerkungen
- <title>

Schema-Spezifikation

Ein eigenes Schema definieren, heißt:

- Bestimme Module auswählen
- In Modulen bestimmte Elemente auswählen
- Für Elemente deren Attribute bestimmten

Schema-Spezifikation

- Ein TEI-Dokument, das eine `<schemaSpec>` enthält, wird ODD genannt (One Document Does it all)
- Hieraus können Schema-Dateien (RelaxNG, W3C Schema, etc.) und eine Dokumentation generiert werden

Wie kann man eine Auswahl treffen?

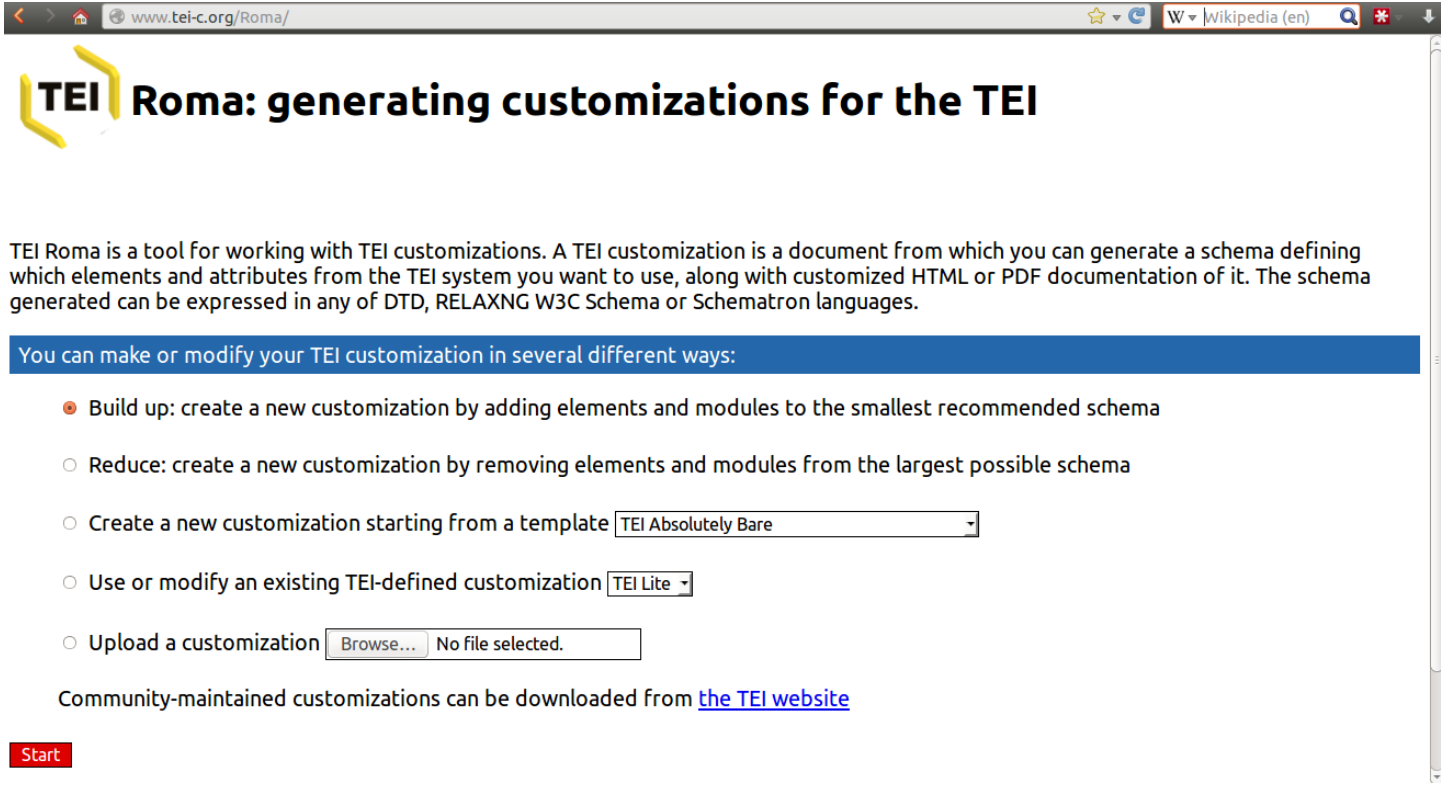
Möglichkeiten

- Alle Module und Elemente verwenden (keine gute Idee)
- Vordefinierte Teilmengen verwenden (TEI Lite, etc.)
- Selbst Module und Elemente auswählen

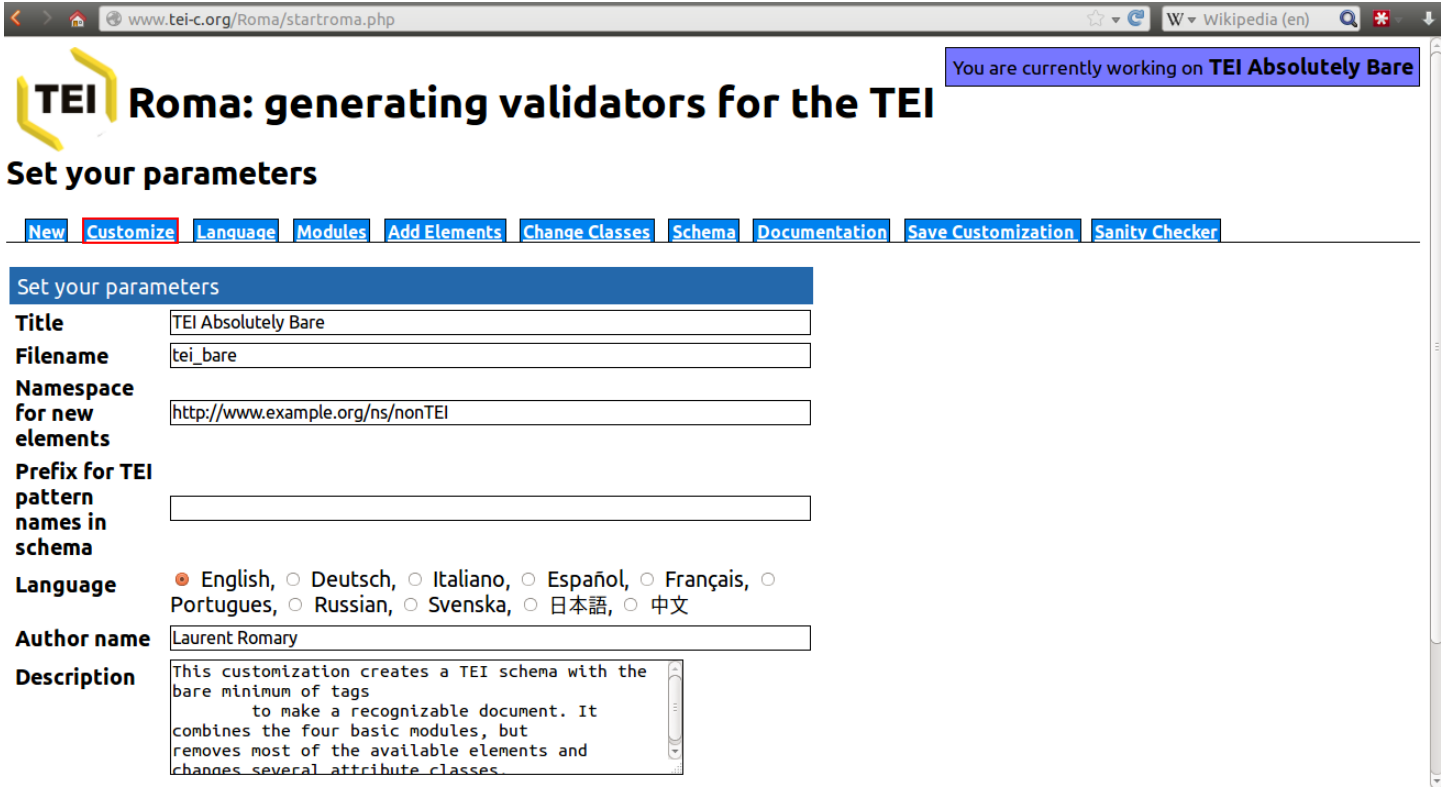
Konkrete Umsetzung

- Entweder selbst ein Schema schreiben (Experten)
- „Roma" verwenden: Web-Frontend für Schema-Generator: http://www.tei-c.org/Roma/

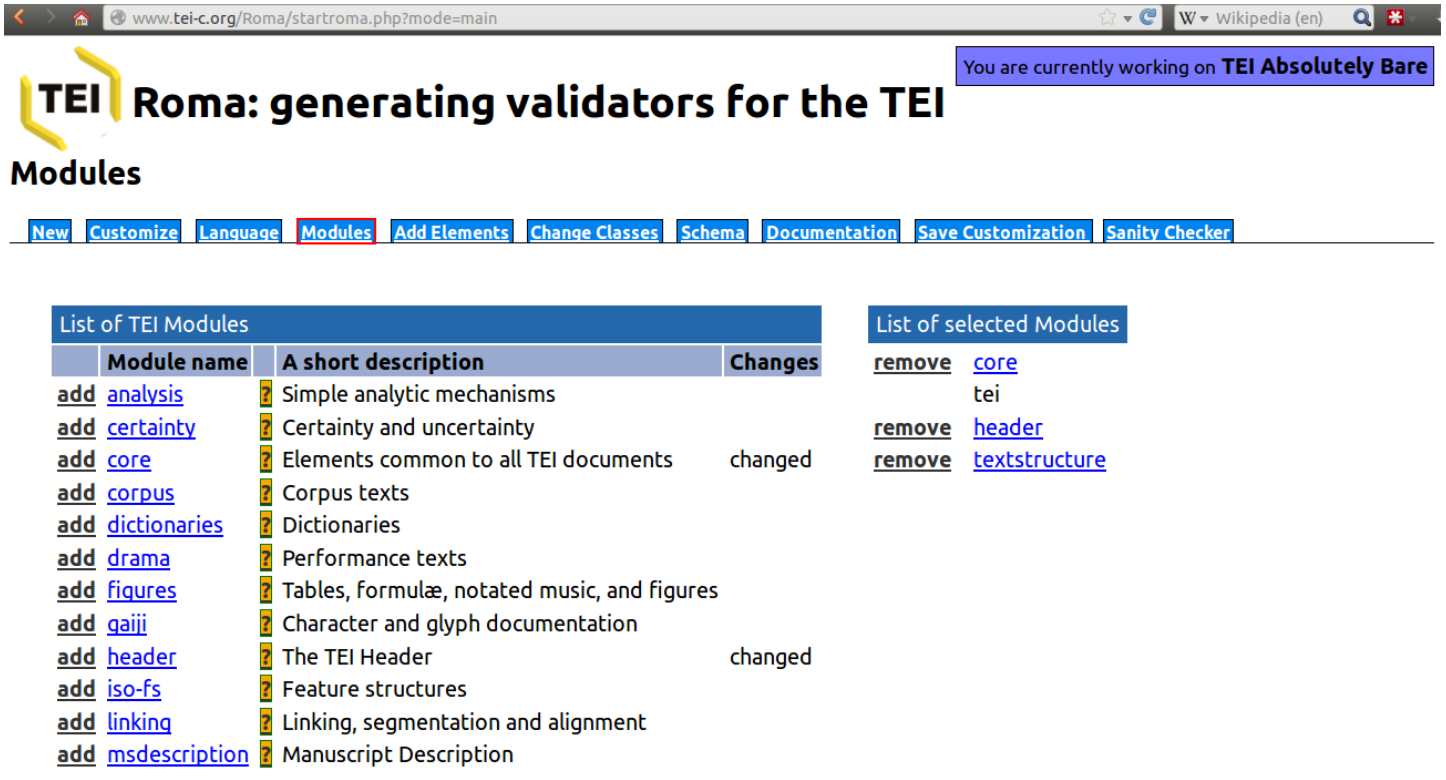
Roma (1): Basis auswählen



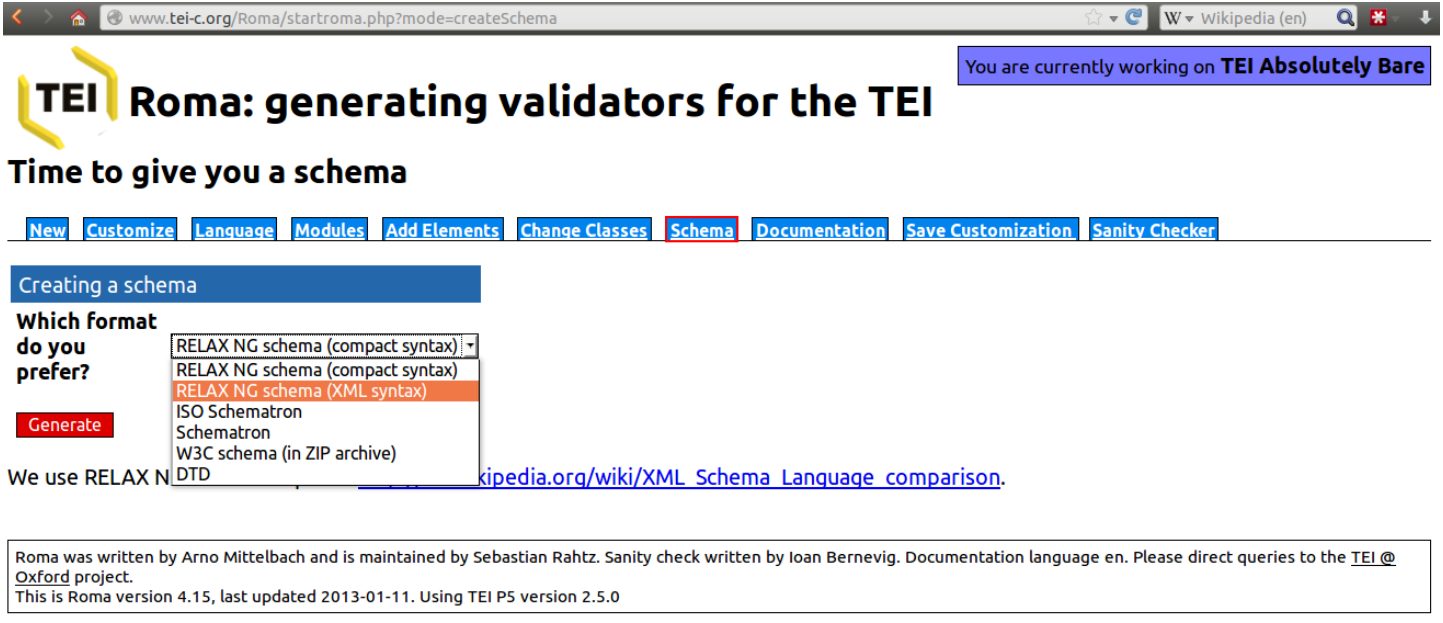
Roma (2): Metadaten eingeben



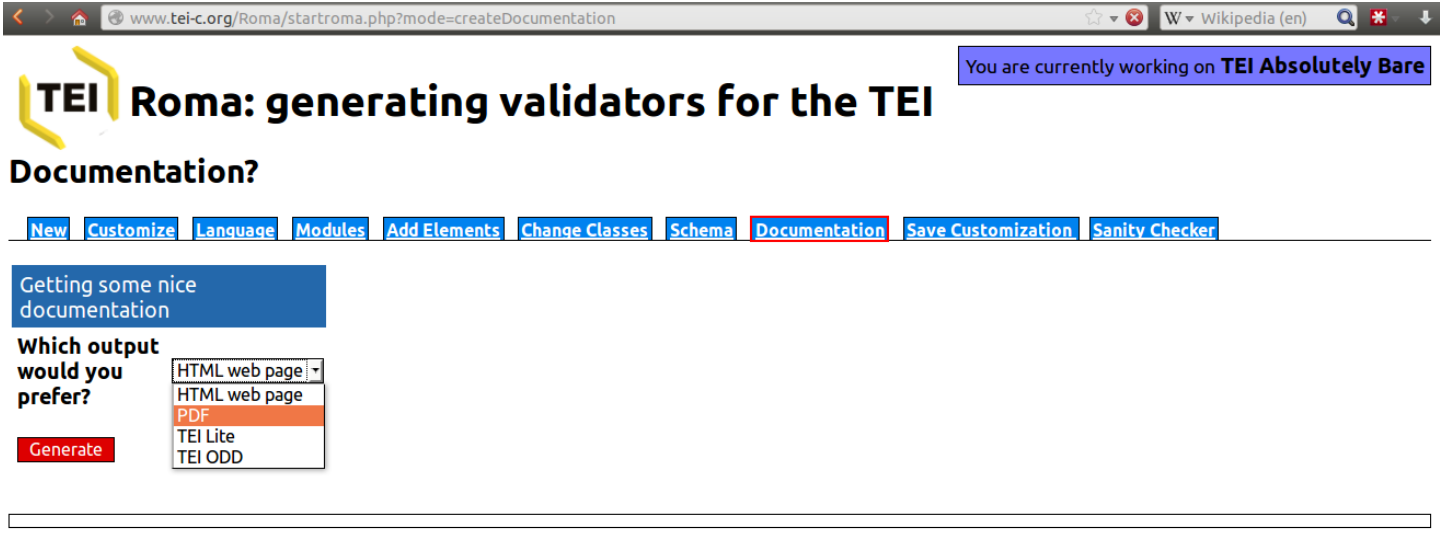
Roma (3): Module und Elemente



Roma (4): Schema generieren



Roma (5): Dokumentation speichern



2. Etwas mehr Informationen zum Klassen-System von TEI

Zugrundeliegende <schemaSpec>

```
...
<schemaSpec ident="tei_bare" start="TEI">
  <moduleRef key="core"/>
  <moduleRef key="tei"/>
  <moduleRef key="header"/>
  <moduleRef key="textstructure"/>
  <elementSpec ident="abbr" mode="delete" module="core"/>
  <elementSpec ident="add" mode="delete" module="core"/>
  <!-- ... -->
  <elementSpec ident="trailer" mode="delete"
  module="textstructure"/>
  <elementSpec ident="title" mode="change" module="core">
    <attList>
      <attDef ident="type" mode="delete"/>
    </attList>
  </elementSpec>
  <!-- ... -->
</schemaSpec>
...
```

Das System der TEI-Klassen („classes”)

Warum ein Klassen-System?

- TEI hat über 500 Elemente
- Klassen schaffen Übersicht, Modularität, machen Änderungen einfacher

Welche Klassen gibt es?

- Attribut-Klassen: alle Elemente in der Klasse haben die gleichen Attribute
- Modell-Klassen: die Elemente haben den gleichen Kontext

Außerdem

- Klassen können andere Klassen beinhalten
- Ein Element kann verschiedenen Klassen angehören
- Die Modul-Zugehörigkeit ist davon unabhängig

Attribut-Klassen

Allgemeines

- Atribut-Klassen haben Namen, die mit „att.” beginnen, bspw.: „att.named”
- Jede Attribut-Klasse beinhaltet ein Set von Attributen
  - „att.named” => @key und @ref;
  - „att-typed” => @type und @subtype.

Also

- Soll ein Element das Attribut @type haben, kann man es der Klasse „att.typed” hinzufügen

Eine wichtige Attribut-Klasse: „att.global”

Alle Elemente sind Mitglieder der Klasse „att.global”

Diese Klasse enthält:

- @xml:id - „unique identifier”
- @xml:lang – die Sprache des Inhalts des Elements
- @n – eine Nummer oder ein Kürzel für ein Element
- @rend – wie der Inhalt des Elements im Quelldokument aussah

Modell-Klassen

Modell-Klassen definieren Kontexte

- Sie enthalten Elemente, die im gleichen Kontext erlaubt sind
- Die Namen von Klassen beruhen auf einem „like” oder „part”-Suffix
- „model.biblLike.class” enthält Element, die sich „like” <bibl> verhalten

Beispiel

- Ein Element das immer dort erlaubt sein soll, wo auch <bibl> erlaubt ist, kann der Klasse „model.biblLike.class” hinzugefügt werden
- Alle Elemente, die in <bibl> vorkommen können, gehören der Klasse „model.biblPart.class” an.

Modell-Klassen: Basics

Es gibt drei „base model classes”

- `divisions` – Grobstruktur von Texten; Bsp.:
- `chunks` – Elemente, die in „divisions” vorkommen; Bsp.: <p>
- `„phrase-level elements”` - Elemente in „chunks”; Bsp.: <hi>

Außerdem:

- `inter-level elements` – Innerhalb oder zwischen „chunks”: Bsp.: <list>
- `components` – Kleinere Elemente, die direkt in „divisions” vorkommen
- `„model.global”`: Elemente, die überall vorkommen können