Introduction to customising the TEI

TEI @ Oxford

September 2014

Customising the TEI

- How the TEI is constructed
- Making a TEI schema
- Specifying your profile of the TEI
- Generating your own documentation

Every use of the TEI should involve making use of a customisation.

Some terminology

- The TEI encoding scheme defines a set of *elements*
- An element definition specifies:
    - a canonical name (<gi>) for the element, and optionally other names in other languages
    - a canonical description (also possibly translated) of its function
    - a declaration of the *classes* to which it belongs
    - a definition for each of its *attributes*
    - a definition of its *content model* (what can appear inside it)
    - usage examples and notes
- *modules* are used to group together sets of elements
- a TEI *schema* specification (<schemaSpec>) is made by selecting modules or elements and (optionally) modifying their contents
- a TEI document containing a schema specification is called an *ODD* (One Document Does it all)

What is a module?

- A convenient way of grouping together a number of element declarations
- These are usually on a related topic or specific application
- Most chapters of P5 focus on elements drawn from a single module, which that chapter then defines
- A TEI schema can be created by selecting modules and adding or removing elements from them as needed

Which modules exist?

Module name	Req.	Opt.	Chapter
analysis			Simple Analytic Mechanisms
certainty			Certainty and Responsibility
core		X	Elements Available in All TEI Documents
corpus			Language Corpora
dictionaries			Dictionaries
drama			Performance Texts
figures			Tables, Formulae, and Graphics
gaiji			Representation of Non-standard Characters
header		X	The TEI Header
iso-fs			Feature Structures
linking			Linking, Segmentation, and Alignment
msdescription			Manuscript Description
namesdates			Names, Dates, People, and Places
nets			Graphs, Networks, and Trees
spoken			Transcriptions of Speech
tagdocs			Documentation Elements
tei	X		The TEI Infrastructure
textcrit			Critical Apparatus
textstructure		X	Default Text Structure
transcr			Representation of Primary Sources
verse			Verse

How do you choose?

- Just choose everything (not really a good idea)
- The TEI provides a small set of predefined combinations (TEI Lite, TEI Bare...)
- Or you could roll your own (but then you need to know what you're choosing)

Roma
: a web-based application designed to make this process much easier

http://www.tei-c.org/Roma/

How does Roma work

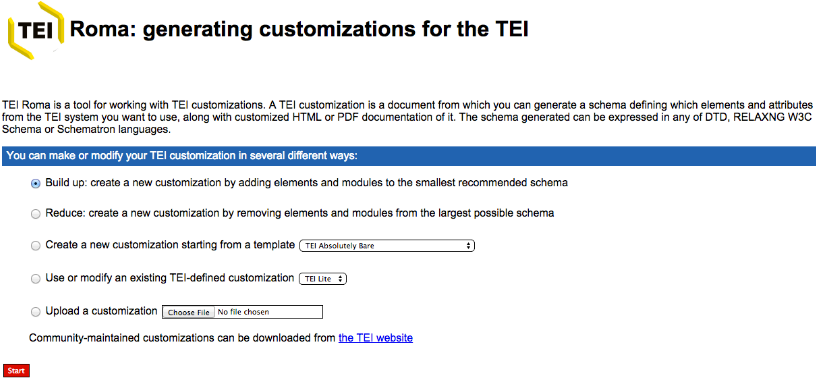
A PHP web application which

1. Queries the TEI source for lists of elements, modules, attributes etc
2. Presents them in a series of forms
3. Generates a TEI ODD specification
4. Sends that ODD to OxGarage, a RESTful web service which uses a set of XSLT transforms to create the desired output

Note that

- OxGarage can be used on its own
- the same transforms can be run within oXygen

Roma: New



TEI Roma: generating customizations for the TEI

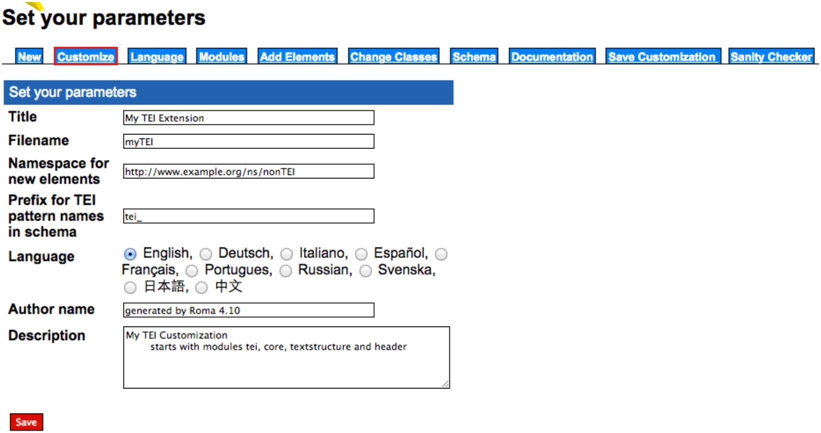
TEI Roma is a tool for working with TEI customizations. A TEI customization is a document from which you can generate a schema defining which elements and attributes from the TEI system you want to use, along with customized HTML or PDF documentation of it. The schema generated can be expressed in any of DTD, RELAXNG W3C Schema or Schematron languages.

You can make or modify your TEI customization in several different ways:

- ● Build up: create a new customization by adding elements and modules to the smallest recommended schema
- ○ Reduce: create a new customization by removing elements and modules from the largest possible schema
- ○ Create a new customization starting from a template: TEI Absolutely Bare
- ○ Use or modify an existing TEI-defined customization: TEI Lite
- ○ Upload a customization: Choose File — No file chosen

Community-maintained customizations can be downloaded from the TEI website

Roma: Customize

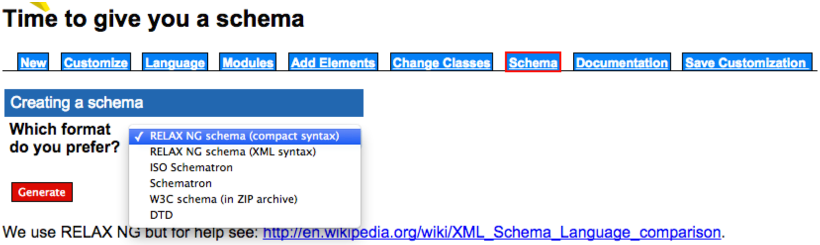


Set your parameters

Set your parameters

Title	My TEI Extension
Filename	myTEI
Namespace for new elements	http://www.example.org/ns/nonTEI
Prefix for TEI pattern names in schema	tei_
Language	● English, ○ Deutsch, ○ Italiano, ○ Español, ○ Français, ○ Portugues, ○ Russian, ○ Svenska, ○ 日本語, ○ 中文
Author name	generated by Roma 4.10
Description	My TEI Customization starts with modules tei, core, textstructure and header

Roma: Schema



Time to give you a schema

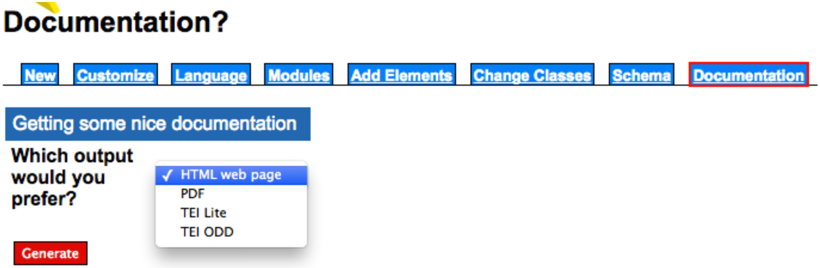
Creating a schema

Which format do you prefer?

✓ RELAX NG schema (compact syntax)
RELAX NG schema (XML syntax)
ISO Schematron
Schematron
W3C schema (in ZIP archive)
DTD
Generate

We use RELAX NG but for help see: http://en.wikipedia.org/wiki/XML_Schema_Language_comparison.

Roma: Documentation



Documentation?

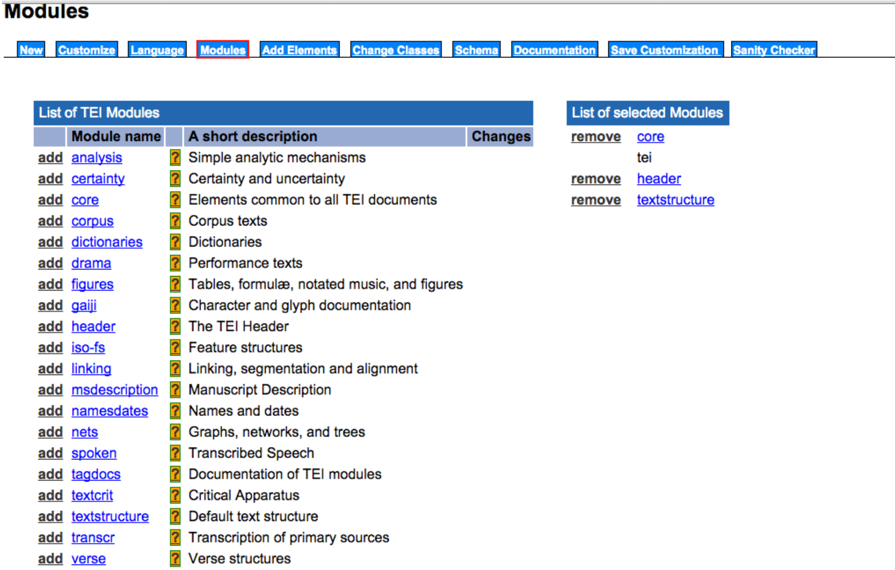
Getting some nice documentation

Which output would you prefer?

Roma provides also an interface to the detail

- The [Modules] tab shows the modules available
- Selecting a module from it shows the elements within that module, and gives you the choice to
    - include all of them (and then remove some)
    - exclude all of them (and then put back the ones you want)
- You can also change an element's attribute list, and the values they permit

Roma: Modules



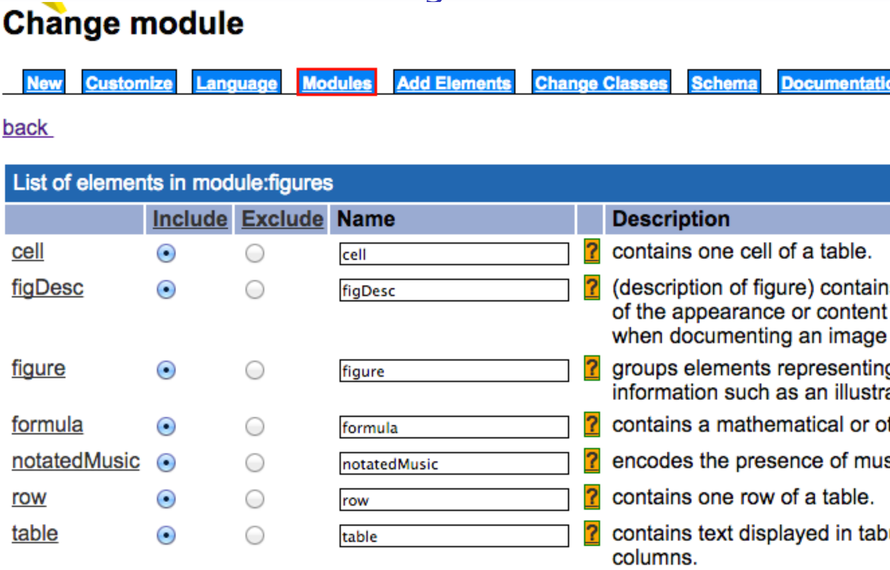
Modules

List of TEI Modules

	Module name		A short description	Changes
add	analysis	?	Simple analytic mechanisms	
add	certainty	?	Certainty and uncertainty	
add	core	?	Elements common to all TEI documents	
add	corpus	?	Corpus texts	
add	dictionaries	?	Dictionaries	
add	drama	?	Performance texts	
add	figures	?	Tables, formulæ, notated music, and figures	
add	gaiji	?	Character and glyph documentation	
add	header	?	The TEI Header	
add	iso-fs	?	Feature structures	
add	linking	?	Linking, segmentation and alignment	
add	msdescription	?	Manuscript Description	
add	namesdates	?	Names and dates	
add	nets	?	Graphs, networks, and trees	
add	spoken	?	Transcribed Speech	
add	tagdocs	?	Documentation of TEI modules	
add	textcrit	?	Critical Apparatus	
add	textstructure	?	Default text structure	
add	transcr	?	Transcription of primary sources	
add	verse	?	Verse structures	

List of selected Modules	
remove	core
	tei
remove	header
remove	textstructure

Roma: Change Module



Change module

back

List of elements in module:figures

	Include	Exclude	Name		Description
cell	●	○	cell	?	contains one cell of a table.
figDesc	●	○	figDesc	?	(description of figure) contain of the appearance or content when documenting an image
figure	●	○	figure	?	groups elements representin information such as an illustra
formula	●	○	formula	?	contains a mathematical or o
notatedMusic	●	○	notatedMusic	?	encodes the presence of mus
row	●	○	row	?	contains one row of a table.
table	●	○	table	?	contains text displayed in tab columns.

Lets go live across to Roma...

What did we just do?

We made an ODD file which contained a schema specification like this:

```
<schemaSpec  ident="myTEI"  docLang="en"
  prefix="tei_"  xml:lang="en">
  <moduleRef  key="core"
    except="abbr add addrLine address analytic author"/>
  <moduleRef  key="tei"/>
  <moduleRef  key="header"/>
  <moduleRef  key="textstructure"/>
  <elementSpec  ident="title"  module="core"
    mode="change">
    <attList>
      <attDef  ident="level"  mode="delete"/>
    </attList>
  </elementSpec>
</schemaSpec>
```

We selected four modules, deleted some elements, and also deleted an attribute.

What else does our customization need?

A simple selection of elements, but also

- we want to allow only certain values for *@type* on <div>
- we may want to create a new element (can you think of something?)

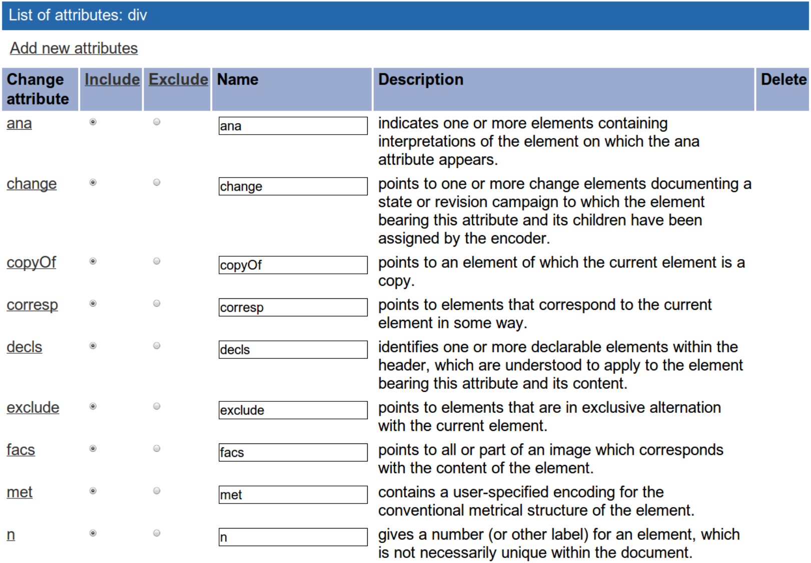
Other constraints are possible — we might want to insist that a `<div type="prose">` contains a paragraph, for example.

The ODD advantage

We can express these constraints in our ODD meta-schema, and then generate a formal schema to enforce them using whichever schema language we like.

- TEI schemas can be generated in
  - ISO RELAX NG language
  - W3C Schema Language
  - XML DTD language
- ODD itself defines an element's content models using a subset of RELAX NG syntax
- Datatypes are defined in terms of W3C datatypes
- Some facilities (e.g. alternation, namespaces) cannot be expressed in DTDs — RELAX NG schema is recommended
- Additional constraints can be expressed in Schematron

Roma: selecting attributes

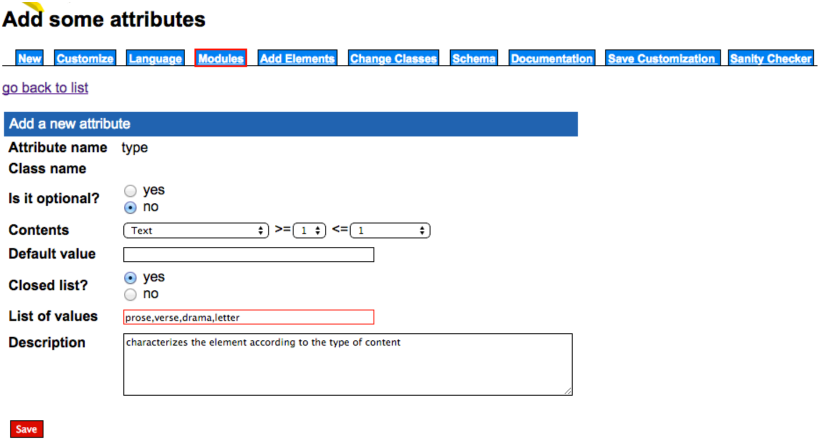


List of attributes: div

Add new attributes

Change attribute	Include	Exclude	Name	Description	Delete
ana	●	○	ana	indicates one or more elements containing interpretations of the element on which the ana attribute appears.	
change	●	○	change	points to one or more change elements documenting a state or revision campaign to which the element bearing this attribute and its children have been assigned by the encoder.	
copyOf	●	○	copyOf	points to an element of which the current element is a copy.	
corresp	●	○	corresp	points to elements that correspond to the current element in some way.	
decls	●	○	decls	identifies one or more declarable elements within the header, which are understood to apply to the element bearing this attribute and its content.	
exclude	●	○	exclude	points to elements that are in exclusive alternation with the current element.	
facs	●	○	facs	points to all or part of an image which corresponds with the content of the element.	
met	●	○	met	contains a user-specified encoding for the conventional metrical structure of the element.	
n	●	○	n	gives a number (or other label) for an element, which is not necessarily unique within the document.	

Roma: constraining attribute values



Add some attributes

go back to list

Add a new attribute

Attribute name	type
Class name	
Is it optional?	○ yes ● no
Contents	Text >= 1 <= 1
Default value	
Closed list?	● yes ○ no
List of values	prose,verse,drama,letter
Description	characterizes the element according to the type of content

What did we just do?

Our ODD now includes something like this:

```
<elementSpec ident="div"
 module="textstructure" mode="change">
 <attList>
  <attDef ident="type" mode="change"
   usage="req">
   <desc>characterizes the element according to the type of content</desc>
   <valList type="closed" mode="replace">
    <valItem ident="prose"/>
    <valItem ident="verse"/>
    <valItem ident="drama"/>
    <valItem ident="letter"/>
   </valList>
  </attDef>
 </attList>
</elementSpec>
```

You can also document attribute values in ODD, but Roma does not support this in its interface:

```
<valItem ident="verse">
 <gloss>contains (parts of ) a poem</gloss>
</valItem>
```

Defining a new element

When defining a new element, we need to consider

- its name and description
- what attributes it can carry
- what it can contain
- where it can appear in a document

The TEI class system helps us answer all these questions (except the first).

The TEI Class System

- The TEI distinguishes over 500 elements,
- Having these organised into classes aids comprehension, modularity, and modification.
- *Attribute class*: the members share common attributes
- *Model class*: they can appear in the same locations (and are often semantically related)
- Classes may contain other classes
- An element can be a member of any number of classes, irrespective of the module it belongs to.

Attribute Classes

- Attribute classes are given (usually adjectival) names beginning with att.; e.g. *att.naming*, *att.typed*
- all members of `att.naming` inherit from it attributes *@key* and *@ref*; all members of `att.typed` inherit from it *@type* and *@subtype*
- If we want an element to carry the *@type* attribute, therefore, we add the element to the `att.typed` class, rather than define those attributes explicitly.

A very important attribute class: att.global

Elements should usually be made members of att.global; this class provides, among others:

@xml:id a unique identifier

@xml:lang the language of the element content

@n a number or name for an element

@rend how the element in question was rendered or presented in the source text.

Model Classes

- Model classes contain groups of elements which are allowed in the same place. e.g. if you are adding an element which is wanted wherever the <bibl> is allowed, add it to the model.biblLike class
- Model classes are usually named with a Like or Part suffix:
  - members of model.pLike are all things that ‘behave like’ paragraphs, and are permitted in the same places as paragraphs
  - members of model.pPart are all things which can appear *within* paragraphs. This class is subdivided into
    - model.pPart.edit elements for simple editorial intervention such as <corr>, <del> etc.
    - model.pPart.data‘data-like’ elements such as <name>, <num>, <date> etc.
    - model.pPart.msdesc extra elements for manuscript description such as <seal> or <origPlace>

Basic Model Class Structure

There are three generally recognized classes of element:

- divisions high level major divisions of texts
- chunks elements such as paragraphs appearing within texts or divisions, but not within other chunks
- phrase-level elements elements such as highlighted phrases which can occur only within chunks

There are also:

- inter-level elements elements such as lists which can appear either in or between chunks
- components elements which can appear directly within texts or text divisions

A special class, model.global, is for elements that can appear *anywhere* inside a text — at any hierarchic level.

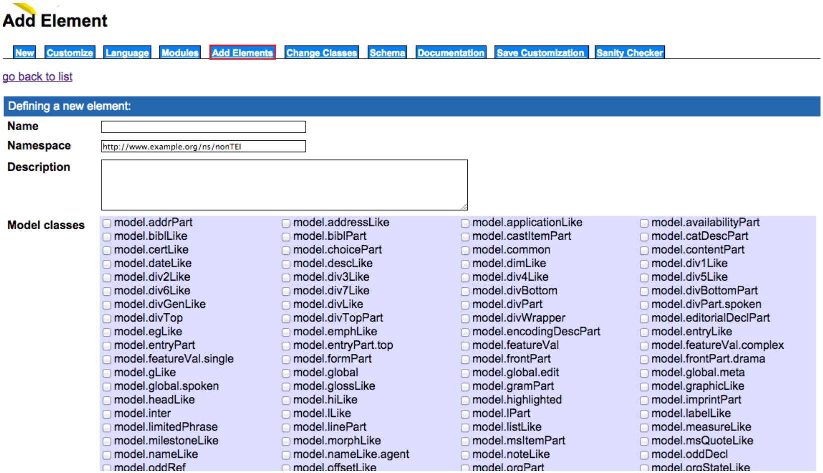
Defining a new element

- What other elements is it like?
- What other elements can contain it?
- What can it contain?

Conclusions:

- What classes do we make it a member of?
- What content model do we select (which classes can appear inside it)?

Roma: Defining a new element



Add Element

go back to list

Defining a new element:

Name

Namespace http://www.example.org/ns/nonTEI

Description

Model classes

□ model.addrPart □ model.addressLike □ model.applicationLike □ model.availabilityPart
□ model.biblLike □ model.biblPart □ model.castItemPart □ model.catDescPart
□ model.certLike □ model.choicePart □ model.common □ model.contentPart
□ model.dateLike □ model.descLike □ model.dimLike □ model.div1Like
□ model.div2Like □ model.div3Like □ model.div4Like □ model.div5Like
□ model.div6Like □ model.div7Like □ model.divBottom □ model.divBottomPart
□ model.divGenLike □ model.divLike □ model.divPart □ model.divPart.spoken
□ model.divTop □ model.divTopPart □ model.divWrapper □ model.editorialDeclPart
□ model.egLike □ model.emphLike □ model.encodingDescPart □ model.entryLike
□ model.entryPart □ model.entryPart.top □ model.featureVal □ model.featureVal.complex
□ model.featureVal.single □ model.formPart □ model.frontPart □ model.frontPart.drama
□ model.gLike □ model.global □ model.global.edit □ model.global.meta
□ model.global.spoken □ model.glossLike □ model.gramPart □ model.graphicLike
□ model.headLike □ model.hiLike □ model.highlighted □ model.imprintPart
□ model.inter □ model.lLike □ model.lPart □ model.labelLike
□ model.limitedPhrase □ model.linePart □ model.listLike □ model.measureLike
□ model.milestoneLike □ model.morphLike □ model.msItemPart □ model.msQuoteLike
□ model.nameLike □ model.nameLike.agent □ model.noteLike □ model.oddDecl
□ model.oddRef □ model.offsetLike □ model.orgPart □ model.orgStateLike

Defining a content model

There are *shortcuts* (macros) for some very common content models:

macro.paraContent content of paragraphs and similar elements

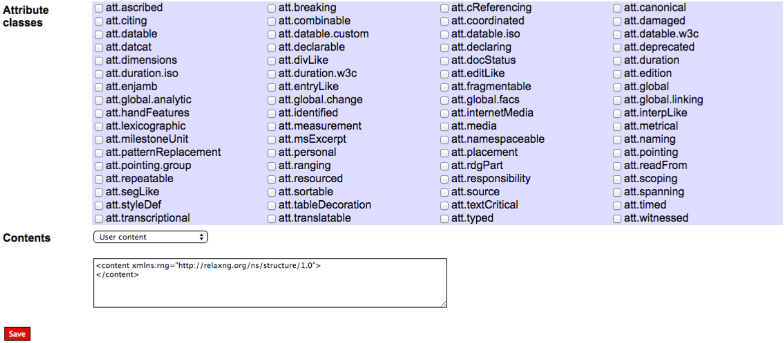
macro.limitedContent content of prose elements that are not used for transcription of extant materials

macro.phraseSeq a sequence of character data and phrase-level elements

macro.phraseSeq.limited a sequence of character data and those phrase-level elements that are not typically used for transcribing documents

macro.specialPara for elements which may contain a series of component-level elements or a series of phrase-level and inter-level elements

Roma: Defining a new element 2



Attribute classes

□ att.ascribed □ att.breaking □ att.cReferencing □ att.canonical
□ att.citing □ att.combinable □ att.coordinated □ att.damaged
□ att.datable □ att.datable.custom □ att.datable.iso □ att.datable.w3c
□ att.datcat □ att.declarable □ att.declaring □ att.deprecated
□ att.dimensions □ att.divLike □ att.docStatus □ att.duration
□ att.duration.iso □ att.duration.w3c □ att.editLike □ att.edition
□ att.enjamb □ att.entryLike □ att.fragmentable □ att.global
□ att.global.analytic □ att.global.change □ att.global.facs □ att.global.linking
□ att.handFeatures □ att.identified □ att.internetMedia □ att.interpLike
□ att.lexicographic □ att.measurement □ att.media □ att.metrical
□ att.milestoneUnit □ att.msExcerpt □ att.namespaceable □ att.naming
□ att.patternReplacement □ att.personal □ att.placement □ att.pointing
□ att.pointing.group □ att.ranging □ att.rdgPart □ att.readFrom
□ att.repeatable □ att.resourced □ att.responsibility □ att.scoping
□ att.segLike □ att.sortable □ att.source □ att.spanning
□ att.styleDef □ att.tableDecoration □ att.textCritical □ att.timed
□ att.transcriptional □ att.translatable □ att.typed □ att.witnessed

Contents User content

```
<content xmlns:rng="http://relaxng.org/ns/structure/1.0">
</content>
```

What did we just do?

We added a new element specification to our ODD, like this:

```
<elementSpec ident="something"
 ns="http://www.example.org/ns/nonTEI" mode="add">
 <desc>contains something division-like, containing
   paragraph-like elements.</desc>
 <classes>
  <memberOf key="model.divPart"/>
  <memberOf key="att.typed"/>
 </classes>
 <content>
  <rng:oneOrMore>
   <rng:ref name="model.pLike"/>
  </rng:oneOrMore>
 </content>
</elementSpec>
```

Note that this new element is *not* in the TEI namespace. It belongs to this specific project only!

Other kinds of constraints

- You can also constrain the content of an element or the value of an attribute to be of a particular *datatype* (for example, to insist that the @*when* attribute of the element <date> contains only a date)
- This can be done by using one of a set of predefined *macros* to define the content. Examples include

data.word	a single word or token
data.name	an XML Name
data.enumerated	a single XML name taken from a documented list
data.temporal.w3c	a W3C date
data.truthValue	a truth value (true/false)
data.language	a human language
data.sex	human or animal sex

- Or you can define a more complex constraint, e.g. using Schematron

Next

That is a quick look at some of the basic things one can do with the TEI ODD language, and the Roma web tool.

Now let's do an exercise where we try out customising the TEI!