

# On-the-fly Validation of XML Markup Languages using off-the-shelf Tools



Mikko Saesmaa  
Pekka Kilpeläinen  
Dept of Computer Science  
University of Kuopio, Finland

## The Talk in a Nutshell

- How to support continuous validation in an XML editor...
  - easily
  - efficiently
  - against different schema languages?
    - » DTD, XML Schema, Relax NG; ..., NVDL for compound docs
- Lesson: straightforward application of Java-XML APIs is effective, and quite efficient, too

EML'07 Montreal

On-the-fly Validation of XML

2

## Intended Audience

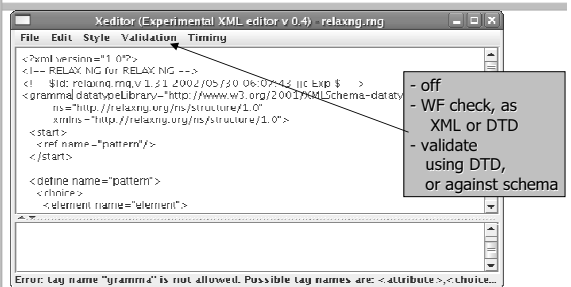
- Suitable for Java/XML developers
  - emphasis on practical use of technology
- EXTREME enough?
  - Nothing extraordinary; some ideas non-obvious
- Why Java?
  - built-in XML and GUI facilities (JAXP + Swing)
  - general and portable

EML'07 Montreal

On-the-fly Validation of XML

3

## Look & Feel of "Xeditor"



EML'07 Montreal

On-the-fly Validation of XML

4

## An Experimental Editor

- Not production quality (yet)
  - UI unfinished
  - useful editor functionality missing
  - slightly unstable
- Purpose to experiment with XML technology
  - examine, learn, and explain

EML'07 Montreal

On-the-fly Validation of XML

5

## Background and Related Work

- Travis, <TAG> 1999: "Real-time XML Editor"
  - > Architag XRay2
- Oxygen, XMLMind, XMLSpy, ...
  - internal solutions of commercial editors?
- Clark's nXML (on GNU Emacs)
  - ~17,000 lines of Emacs Lisp (vs. ~1000 of ours)
- DB research on **incremental** XML validation (Barbosa et al. 2004 & 2006, Balmin et al. 2004)

EML'07 Montreal

On-the-fly Validation of XML

6

## Architectural solutions of Xeditor

- Separation of Concerns: Editor (Swing) ignorant of XML, **XMLHandler** (JAXP) ignorant of editing
- After each edit, the doc is re-parsed completely
- Pro: modularity and simplicity
- Cons:
  - limited functionality (e.g. markup completion and syntax highlighting)
  - some inefficiency, but not too bad

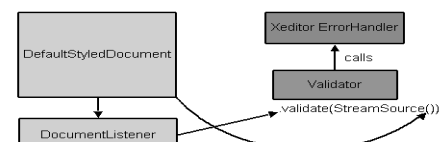
EML'07 Montreal

On-the-fly Validation of XML

7

## Basic Technicalities

- Event-driven document checking/validation
  - modifications caught by a Swing **DocumentListener**
  - **errors caught as SAX parse exceptions**

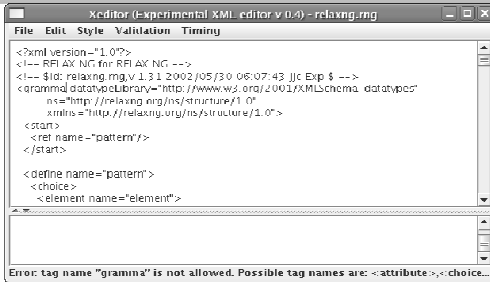


EML'07 Montreal

On-the-fly Validation of XML

8

## Error reporting in Xeditor



EML' 07 Montreal

On-the-fly Validation of XML

9

## Basic Technicalities (2)

- JAXP is based on a Factory Design Pattern
  - Initialization of a Factory selects appropriate implementation of a parser, schema language, XSLT transformer, ... (Implementation-independence, *pluggability*)
  - > *extensibility* by new schema languages

EML' 07 Montreal

On-the-fly Validation of XML

10

## Basic Technicalities (3)

- Well-formedness checking and DTD validation based on JAXP SAX interfaces:
  - **SAXParserFactory** → **SAXParser**
  - Validating/non-validating parser selected based on editing mode
  - SAX instead of DOM relevant for efficiency
  - Only errors are observed; other events ignored
- Schema-based validation using JAXP Validators (later)

EML' 07 Montreal

On-the-fly Validation of XML

11

## Efficiency Concerns

- Rule of thumb: access via memory much faster than via disk, which again much faster than over network
- Document passed to parser/validator as an in-memory stream
  - > normally no observable delays
- Could cache external entities, like DTDs, too

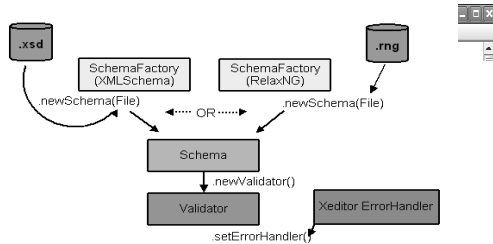
EML' 07 Montreal

On-the-fly Validation of XML

12

## Different Schemas and Schema Languages

- A Validator created when the user selects Schema



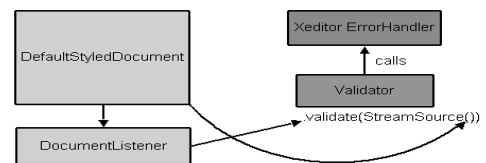
EML' 07 Montreal

On-the-fly Validation of XML

13

## Validation against the Schema

- As already shown:



EML' 07 Montreal

On-the-fly Validation of XML

14

## Editing Schema Documents

- Public schema files for schema languages used for validation (as external resource files)
  - **.xsd** or **XMLSchema.xsd** for XML Schema
  - **relaxng.rng** for Relax NG
- Pro: uniformity; efficiency
- Con: imprecision (vs validating by appropriate schema compiler)

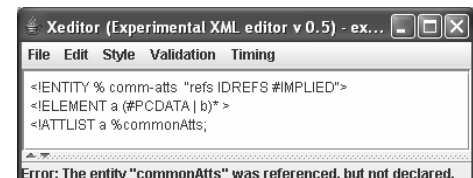
EML' 07 Montreal

On-the-fly Validation of XML

15

## Editing DTDs

- How to check the **non-XML** syntax of DTDs easily?



EML' 07 Montreal

On-the-fly Validation of XML

16

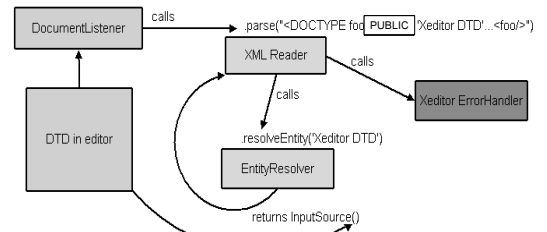
## Editing DTDs (2)

- Soln 1: Wrap the DTD in the prolog of a dummy doc
  - how to check stand-alone DTDs (“external subsets”)?
- Soln 2: Parse a fixed dummy doc that refers to the DTD:

```
<!DOCTYPE foo PUBLIC "Xeditor DTD" "IN-MEMORY"
[<!ELEMENT foo EMPTY> ]> <foo/>
```

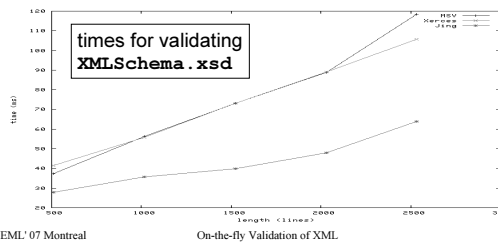
randomize “foo”,  
to avoid conflict  
with actual DTD

## Editing DTDs (3)

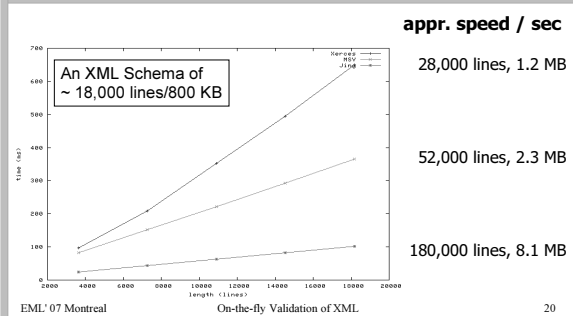


## Efficiency and Scalability

- Is brute-force re-validation too inefficient?
- No: Delays normally unnoticeable



## Validating a Larger Instance



## Conclusions

- Immediate validation against schemas in different languages (DTD, XML Schema, Relax NG) easy to support
- Efficiency of brute-force application sufficient for moderate-sized documents

## Further Work

- Potential application: teaching of XML
- Practical enhancements
  - markup completion, highlighting etc.
- Incremental validation
  - to remove dependency of document length
  - How to apply/modify standard interfaces?
- Thank you! Questions? Comments?