XML- eXtensible Markup Language

mariaiuliana.dascalu@gmail.com

University POLITEHNICA of Bucharest
Department of Engineering in Foreign Languages

Agenda

- XML Generalities
- DTD(Document Type Declaration)

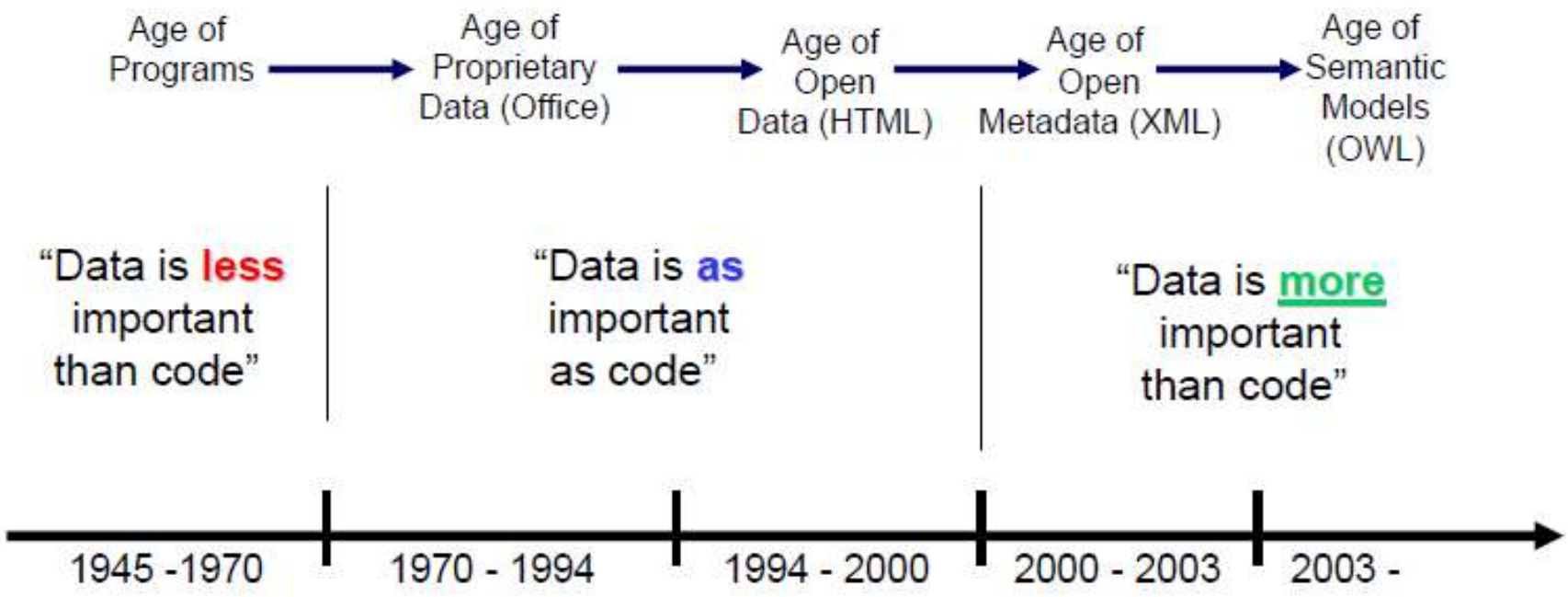
Questions

- How do we model/represent data?
- What kind of data model do we choose for:
  - stocking heterogeneous data coming from multiple sources?
  - information which evolve in time?
  - representing natural language?

How do we model/represent data?

Data are more important than applications…

The evolution in time of DATA:



Problems of sharing data…

- Syntactic sharing problem:
  - Finding a common medium for communication
- Semantic sharing problem:
  - Finding a mutual encoding of concepts within a common medium

Necessities

- The data we want to model might be unknown and unbound
- We don’t have a-priori a common vocabulary/schema
- The data have to be self-explanatory
- The model has to be suited for the existing navigational architectures based on hypertext
- The model has to provide support for URI

Solution

- Use of XML- eXtensible Markup Language
- Markup: annotation, codification; e.g.: punctuation marks, delimiters used in source code
- Markup language: set of markup conventions used for coding information; specifies all the mandatory markups, the way in which these markups are identified and structured

XML - Generalities

- markup language designed to carry data, not to display data (like HTML)
- simplified descendant of Standard Generalized Markup Language (ISO 8879:1986 SGML)
- W3C standard for data exchange (1998, 2000, 2004, 2006, 2008):

  http://www.w3.org/TR/REC-xml/

  http://www.w3.org/XML/
- input and output data of applications can be described using XML
- XML is a cross-platform, software and hardware independent tool for transmitting information
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data

XML Can be Used to Create New Languages…

- WAP (Wireless Application Protocol) and WML (Wireless Markup Language)
- WML, used to markup Internet applications for handheld devices like mobile phones
- WSDL for describing available web services
- RSS languages for news feeds
- RDF and OWL for describing resources and ontology

HTML vs XML

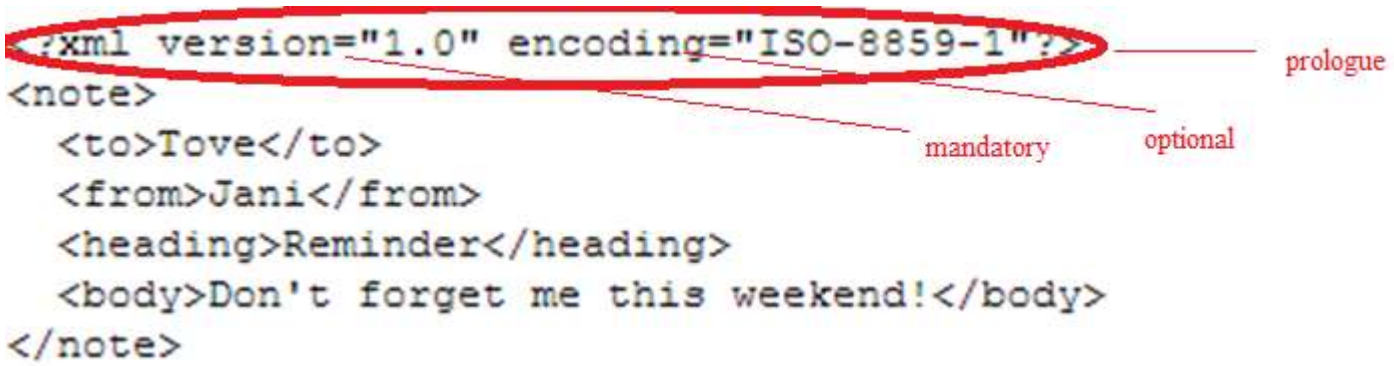
- In HTML docs: same/predefined tags
- In XML docs: completely different, user-defined tags

- HTML tags/annotations define display: colour, lists, font…
- XML meta markup language: language for defining markup languages

- XML is well-formed!!!
- XML is case-sensitive!!!

An XML document consists of…

- a prologue and a number of elements




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- XML docs form a tree structure

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

XML Elements

- An element consists of:
  - an opening tag
  - the content
  - a closing tag
- Tag names can be chosen almost freely.
- Names cannot contain spaces.
- The first character of a name must be a letter or an underscore.
- No name may begin with the string “xml” in any combination of cases (e.g. “Xml”, “xML”).
- Content may be text, or other elements, or nothing:

```
<lecturer/>

<lecturer>
    <name>Susana Caravana</name>
    <phone> +61 − 7 − 3875 507 </phone>
</lecturer>
```

XML Attributes

- An empty element is not necessarily meaningless: it may have some properties in terms of attributes

- An attribute is a name-value pair inside the opening tag of an element:

<lecturer name=“Susana Caravana" phone="+61 – 7 – 3875 507"/>

- Elements may contain attributes. If an attribute is present, it must have a value, even if it is an empty string "".

XML Attributes: An Example

```
<order orderNo="23456" customer="John
  Smith" date="October 15, 2002">
  <item itemNo="a528" quantity="1"/>
  <item itemNo="c817" quantity="3"/>
</order>
```

The Same Example without Attributes

```
<order>
   <orderNo>23456</orderNo>
   <customer>John Smith</customer>
   <date>October 15, 2002</date>
    <item>
         <itemNo>a528</itemNo>
         <quantity>1</quantity>
    </item>
    <item>
         <itemNo>c817</itemNo>
         <quantity>3</quantity>
    </item>
</order>
```

XML Elements vs Attributes

- Attributes can be replaced by elements
- When to use elements and when attributes is a matter of taste
- Attributes cannot be nested, elements can
- Attributes cannot contain multiple values , elements can
- Attributes cannot contain tree structures , elements can
- Attributes are not easily expandable, for future changes
- Both elements and attributes are case-sensitive

```
<img src="..." />
≠
<img SRC="..." />
```

The Tree Model of XML Docs

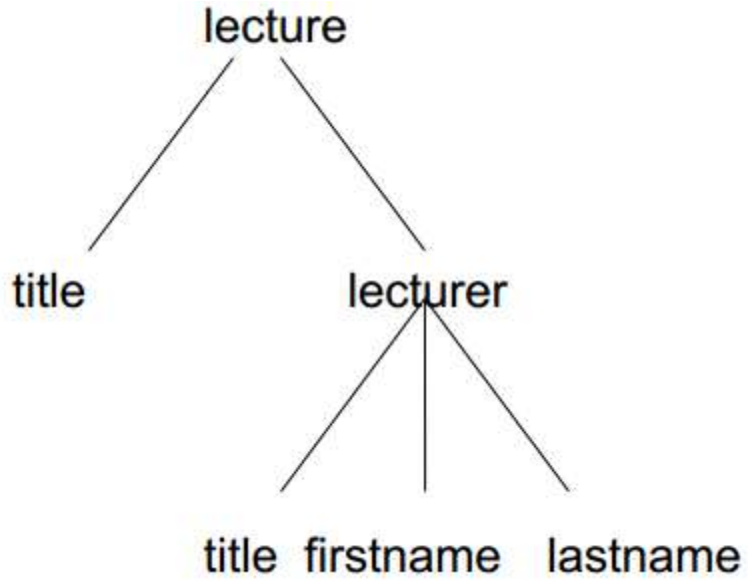
- The tree representation of an XML document is an ordered labelled tree:
  - There is exactly one root
  - There are no cycles
  - Each non-root node has exactly one parent
  - Each node has a label.
  - The order of elements is important
  - … but the order of attributes is not important

XML Tags

- Can be nested

```
<lecture>
        <title> COMS4995 </title>
        <lecturer>
        <title> Dr.</title>
                <firstname>Knarig </firstName>
                <lastname> Arabshian </lastName>
        </lecturer>
</lecture>
```

- Have a tree structure



Comments, White Spaces and Entity References

- <!-- all the comments go in here -->
- White-space is Preserved in XML (not truncated like in HTML)
- Some characters have a special meaning in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Spot the error!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <note date=12/11/2002>
      <to>Tove</to>
      <from>Jani</from>
  </note>
```

Spot the error!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <note date="12/11/2002">
      <to>Tove</to>
      <from>Jani</from>
  </note>
```

Exercise 1 (1 pt: a- 0.25 pts, b-0.75 pts)

a) Is the document below well-formed? If not, make the necessary corrections so that it becomes well-formed. The XML document describes a catalog of courses.

```
<?xml version="1.0" encoding="utf-8" ?>
<catalogue>
    <course mnemonic="INFO-H-509">
        <title>Technologies XML
        <professor email="stijn.vansummeren@ulb.ac.be">Stijn Vansummeren</prof>
        <assistant email=fpicalau@ulb.ac.be>François Picalausa</assistant>
        <weight>3 <unit type="ECTS" /></weight>
    </course>
</catalogue>
<catalogue>
    <course mnemonic="INFO-H-302" <!-- Ce cours ne sera plus donné l'année
    prochaine --> >
        <title>Analyse & conception par object</title>
        <professor email="ezimanyi@ulb.ac.be">Esteban Zimányi
        <assistant email="fservais@ulb.ac.be">Frédéric Servais</assistant>
        < assistant email="sboucher@ulb.ac.be">Serge Boucher</assistant>
        <weight>3 <unit type="ECTS"></weight>
    </course>
</catalogue>
```

b) Complete this document with several elements person, in order to include the following information. You have to use wisely elements and attributes.

Name	Laboratory	Email	Fields of Research
Stijn Vansummeren	WIT	stijn.vansummeren@ulb.ac.be	Systèmes d'information Bases de données scientifiques Web sémantique
Esteban Zimányi	WIT	ezimanyi@ulb.ac.be	Systèmes d'information Entrepôts de données Web sémantique
François Picalausa	WIT	fpicalau@ulb.ac.be	Web sémantique

XML Validators

http://www.w3schools.com/xml/xml_validator.asp

http://www.validome.org/xml/validate/

http://www.w3.org/2001/03/webdata/xsv

XML Viewing

- XML files can be viewed in all major browsers.
- If an erroneous XML file is opened, the browser will report the error.

Document Type Declaration (DOCTYPE)

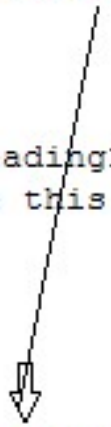
- *which* tags and attributes are allowed,
- *where* they can be placed,
- whether or not they can be *nested* within a given document and
- *what* additional entity definitions are required

- Used to validate an xml document (an XML-alternative of DTD is XML Schema)

External DTD

- <!DOCTYPE root-element SYSTEM "filename">

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```



```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Internal DTD

- <!DOCTYPE root-element [element-declarations]>

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

(View page source for the DTD)

PCDATA vs. CDATA

- PCDATA is parsed character data. PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.
- PCDATA should not contain any &, <, or > characters

- CDATA is character data. CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

Declaring Elements in DTD (1)

- Declaring empty elements <!ELEMENT element-name EMPTY>

Example: <!ELEMENT br EMPTY>

- Elements with any contents <!ELEMENT element-name ANY>

Example: <!ELEMENT note ANY>

- Elements with PCDATA <!ELEMENT element-name (#PCDATA)>

Example: <!ELEMENT from (#PCDATA)>

- Elements with children <!ELEMENT element-name (child1,child2,...)>

Example: <!ELEMENT note (to,from,heading,body)>

- Declaring only 1 occurrence of an element <!ELEMENT element-name (child-name)>

Example: <!ELEMENT note (message)>

- Declaring min 1 occurrence of an element <!ELEMENT element-name (child-name+)>

Example: <!ELEMENT note (message+)>

Declaring Elements in DTD (2)

- Declaring 0 or >0 occurrences of an element <!ELEMENT element-name (child-name*)>

Example: <!ELEMENT note (message*)>

- Declaring 0 or 1 occurrences of an element

<!ELEMENT element-name (child-name?)>

Example: <!ELEMENT note (message?)>

- Declaring either/or an element

Example: <!ELEMENT note (to,from,header,(message|body))>

Declaring Attributes in DTD

- <!ATTLIST element-name attribute-name attribute-type default-value>
- <!ATTLIST element-name attribute-name (en1|en2|..) default-value>

Example: <!ATTLIST payment type CDATA "check">

For <payment type="check" />

- The **attribute-type** can be one of the following: CDATA (character data), ID (unique id), IDREF (the id of another element),…
- The **default-value** can be one of the following: *value,* #REQUIRED (mandatory), #IMPLIED (optional), #FIXED *value* (can’t be changed)

Examples

DTD	XML
<!ATTLIST payment type CDATA "check">	<payment type="check" />
<!ATTLIST payment type (check|cash) "cash">	<payment type="check" /> or <payment type="cash" />
<!ATTLIST contact fax CDATA #IMPLIED>	<contact /> or <contact fax="555-667788" />
<!ATTLIST person number CDATA #REQUIRED>	<person number="5677" />
<!ATTLIST sender company CDATA #FIXED "Microsoft">	<sender company="Microsoft" />

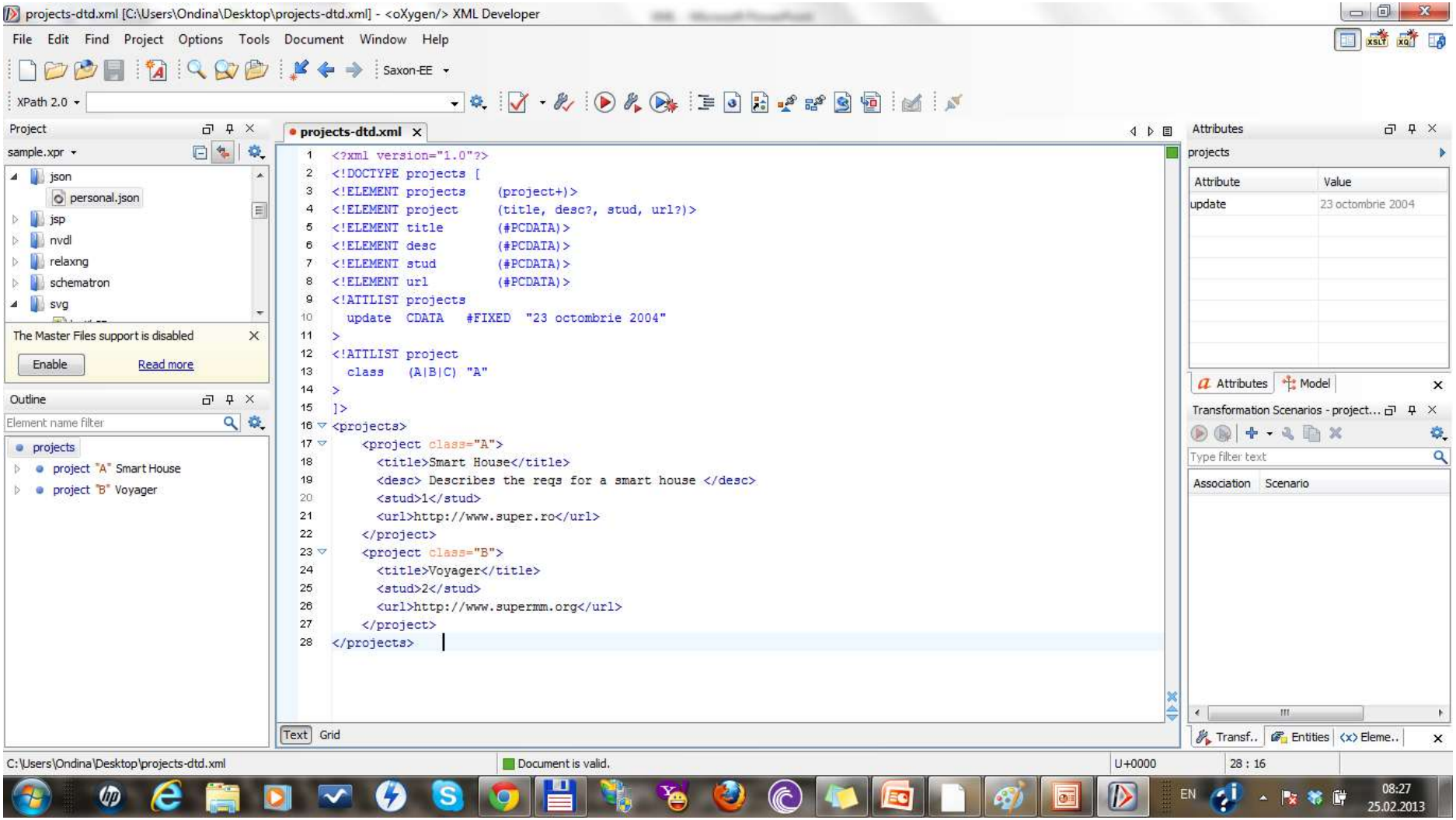
Validators

- http://www.w3schools.com/xml/xml_validator.asp

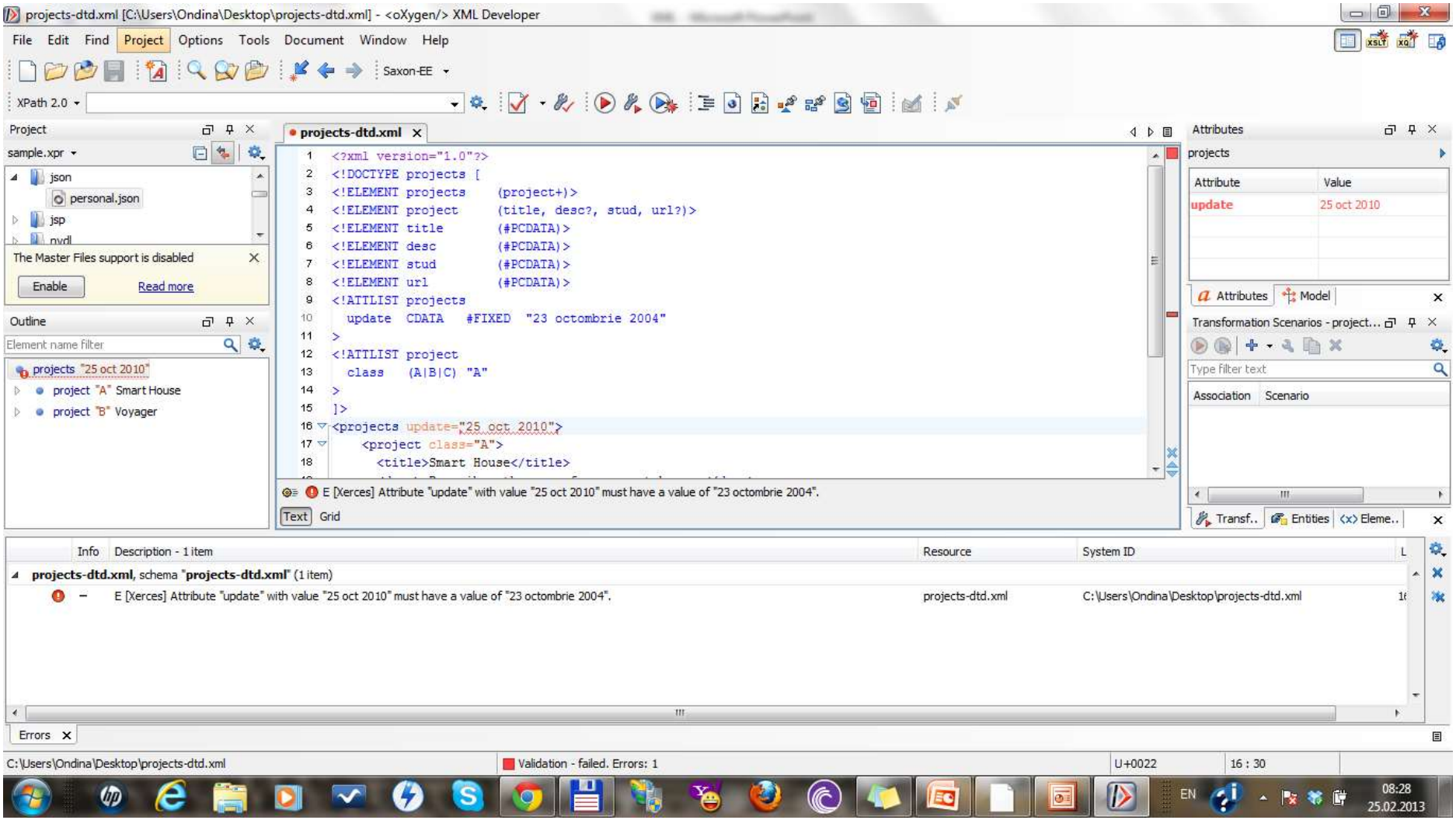
- Oxygen XML Developer:

http://www.oxygenxml.com/download_oxygenxml_developer.html

Valid XML


projects-dtd.xml [C:\Users\Ondina\Desktop\projects-dtd.xml] - <oXygen/> XML Developer
File Edit Find Project Options Tools Document Window Help
Saxon-EE
XPath 2.0
Project
sample.xpr
json
personal.json
jsp
nvdl
relaxng
schematron
svg
The Master Files support is disabled
Enable
Read more
Outline
Element name filter
projects
project "A" Smart House
project "B" Voyager
projects-dtd.xml
1 <?xml version="1.0"?>
2 <!DOCTYPE projects [
3 <!ELEMENT projects (project+)>
4 <!ELEMENT project (title, desc?, stud, url?)>
5 <!ELEMENT title (#PCDATA)>
6 <!ELEMENT desc (#PCDATA)>
7 <!ELEMENT stud (#PCDATA)>
8 <!ELEMENT url (#PCDATA)>
9 <!ATTLIST projects
10 update CDATA #FIXED "23 octombrie 2004"
11 >
12 <!ATTLIST project
13 class (A|B|C) "A"
14 >
15]>
16 <projects>
17 <project class="A">
18 <title>Smart House</title>
19 <desc> Describes the reqs for a smart house </desc>
20 <stud>1</stud>
21 <url>http://www.super.ro</url>
22 </project>
23 <project class="B">
24 <title>Voyager</title>
25 <stud>2</stud>
26 <url>http://www.supermm.org</url>
27 </project>
28 </projects>
Text Grid
Attributes
projects
Attribute Value
update 23 octombrie 2004
Attributes Model
Transformation Scenarios - project...
Type filter text
Association Scenario
Transf.. Entities <x> Eleme..
C:\Users\Ondina\Desktop\projects-dtd.xml
Document is valid.
U+0000
28 : 16
EN
08:27
25.02.2013

Unvalid XML


projects-dtd.xml [C:\Users\Ondina\Desktop\projects-dtd.xml] - <oXygen/> XML Developer
File Edit Find Project Options Tools Document Window Help
Saxon-EE
XPath 2.0
Project
sample.xpr
json
personal.json
jsp
The Master Files support is disabled
Enable
Read more
Outline
Element name filter
projects "25 oct 2010"
project "A" Smart House
project "B" Voyager
projects-dtd.xml
<?xml version="1.0"?>
<!DOCTYPE projects [
<!ELEMENT projects (project+)>
<!ELEMENT project (title, desc?, stud, url?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT desc (#PCDATA)>
<!ELEMENT stud (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST projects
update CDATA #FIXED "23 octombrie 2004"
>
<!ATTLIST project
class (A|B|C) "A"
>
]>
<projects update="25 oct 2010">
<project class="A">
<title>Smart House</title>
E [Xerces] Attribute "update" with value "25 oct 2010" must have a value of "23 octombrie 2004".
Text Grid
Attributes
projects
Attribute Value
update 25 oct 2010
Attributes Model
Transformation Scenarios - project...
Type filter text
Association Scenario
Transf.. Entities <x> Eleme..
Info Description - 1 item Resource System ID
projects-dtd.xml, schema "projects-dtd.xml" (1 item)
E [Xerces] Attribute "update" with value "25 oct 2010" must have a value of "23 octombrie 2004". projects-dtd.xml C:\Users\Ondina\Desktop\projects-dtd.xml
Errors
C:\Users\Ondina\Desktop\projects-dtd.xml
Validation - failed. Errors: 1
U+0022
16 : 30
EN
08:28
25.02.2013