

Web Development Foundations

A Data Model for the Web.
XML Family. XML Processing.

Assoc. Prof. Andrei PANU – profs.info.uaic.ro/andrei.panu/

* many thanks to Prof. Sabin-Corneliu Buraga for the content on these slides

Table of Contents

1. How do we model data
2. What does a markup language represent
3. XML (eXtensible Markup Language)
4. How do we specify XML namespaces
5. How can we process XML documents

How do we model (represent) data?

Preamble

Data must be reused and shared on the Web

open data

“a piece of content or data is open
if anyone is free to use, reuse, and redistribute it”

opendefinition.org

Preamble

What data representation model
can we choose for ...

storing heterogeneous data from multiple sources?
information that evolves over time?
representing natural language?

Preamble

We want to model and process data regarding

poem anthologies

product catalogs of an e-shop

gastronomic recipe repositories

questionnaires

social networks

...

Preamble

Necessities:

**a language able to annotate information
in an explicit manner**

data to be modeled could be practically
unbounded and unknown

no a priori existence of a common vocabulary/schema

Preamble

Necessities:

data should be **self-explanatory**

what does the triple ("Andrei", "Panu", 10281) mean?

Preamble

Necessities:

the adopted model should consider existing navigational architectures, based on hypertext

support for specifying URIs/IRIs

What does a markup language represent?

Preamble – Definitions

Markup – annotation, coding

any action denoting explicit interpretation
of a text (content) fragment

```

$finfo = new finfo(FILEINFO_MIME_TYPE);
if (FALSE === $ext = array_search($finfo->file($_FILES['img']['tmp_name']),
    [ 'jpg' => 'image/jpeg',
      'png' => 'image/png',
      'gif' => 'image/gif',
      'webp' => 'image/webp',
      'svg' => 'image/svg+xml' ], true)) {
    throw new RuntimeException('Upload: format incorrect (see F,
}

// mutam fisierul transferat in directorul cu imagini
// (generam un nume unic pentru fiecare imagine via algoritmul SHA1')
// detalii la https://php.net/manual/en/function.hash.php
$numeImg = sprintf('%s.%s',
    sha1_file($_FILES['img']['tmp_name']), $ext);
if (FALSE === @move_uploaded_file($_FILES['img']['tmp_name'],
    IMGDIR . $numeImg)) {
    throw new RuntimeException('Upload: eroare la salvare.');
```

special
markups

examples:

punctuation marks for written languages (e.g., ¿Vamos?),
delimiters used in source code

Preamble – Definitions

Markup language

set of markup conventions used for data encoding

Preamble – Definitions

Markup language

defines the set of mandatory markups,
the manner of how these markups are identified
and structured by using one or more specifications
(i.e. grammars)

XML

Extensible Markup Language

annotation meta-language

W3C standard (1998, 2000, 2004, 2006, 2008)

www.w3.org/TR/xml/

XML

a technology
+
a family of languages

www.w3.org/XML/

XML – Characterization

Descriptive markups

<para>

<response>

<Person>

<tag>

<meta charset="utf-8" />

<MenuItem>

XML – Characterization

advanced

Types of documents

Document Type Definition (DTD)

formal specification of document types
(components + structure) – i.e. grammar

used to verify the syntactic correctness

XML – Characterization

Data independence

support on every hardware/software platform

XML processors available
for all programming languages

XML – Features

Meta-language

able to define other markup languages

portable

encoding/language independent via Unicode

XML – Features

**Solution for content representation of
the Web resources identified by URI/IRI**

ensuring interoperability (*lingua franca*)

documents are data

XML – Constituents

Prologue (preamble)

Elements

Attributes

Entities


CDATA sections

Processing instructions

XML – Prologue

Declaration that specifies
document version and encoding

```
<?xml version="1.0" encoding="UTF-8" ?>
```



required
attribute



optional
attribute

must appear only once at the beginning of the document

XML – Elements

Element = structural component (textual unit)

XML – Elements

Element = structural component (textual unit)

name – identifies an element
syntax similar to variable identifiers

product

XML – Elements

Syntactically, an element is specified by markups (**tags**) – start tag and end tag

<product>Ping Uinux</product>



start tag



end tag

XML – Elements

Case sensitive

`<markup>` ≠ `<Markup>` ≠ `<MARKUP>`

XML – Elements

An element can have an empty content
(empty element)

```
<product></product>
```

short syntax:

```
<product />
```

XML – Elements

An element can have an empty content
(empty element)

real examples – HTML specification with XML syntax:

```
<br />  
<meta />  
<track />  
<input />
```

XML – Elements

An element can have an empty content

```
<!-- real example: JSX (used by React.js) -->
<Form>
  <Form.Row>
    <Form.Label />
    <Form.Input />
  </Form.Row>
</Form>
```

facebook.github.io/jsx/

XML – Content Models

Structural model

denotes relations between elements:
sequence, hierarchy, grouping, inclusion

XML – Content Models

Elements nested in other elements
(could contain textual data and/or other elements)

```
<product>  
  Ping Uinux is a  
  <obs>multi-colored</obs>  
  mascot which sells  
  <obs>very quickly</obs>.  
</product>
```

<product> (parent node)
includes text and **<obs>**
elements (child nodes)

XML – Content Models

Elements nested in other elements
(could contain textual data and/or other elements)

```
<!-- HTML5 markups respecting the XML conventions -->
<article>
  <section>
    <ul>
      <li><strong>Stagii pe Bune</strong></li>
      <li>Exercism.org</li>
      <li>Code Golf</li>
    </ul>
  </section>
</article>
```

grouping {

XML – Content Models

Elements nested in other elements
(could contain textual data and/or other elements)

```
<!-- HTML5 markups respecting the XML conventions -->
<article>
  <section> } hierarchy
  <ul>
    <li><strong>Stagii pe Bune</strong></li>
sequence { <li>Exercism.org</li>
           <li>Code Golf</li>
  </ul>
  </section>
</article>
```

XML – Content Models



Elements must be correctly closed and nested

```
<div><q>We don't need no education</div></q>
```



wrong

```
<?xml version="1.0" ?>
```



preamble

```
<anthology>
```

```
  <poem>
```

```
    <title>...</title>
```

```
    <stanza>
```

```
      <line>...</line>
```

```
      <line>...</line>
```

```
      ...
```

```
    </stanza>
```

```
  </poem>
```

```
  <!-- more poems... -->
```

```
  <poem>
```

```
    <title />
```

```
  </poem>
```

```
  <poem>
```

```
  </poem>
```

```
</anthology>
```

an XML document modeling
an anthology of poems

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<products>
```

```
  <product>
```

```
    <name>Ping Unix</name>
```

```
    <provider>http://www.penguin.info</provider>
```

```
    <promo>Mascot of the month</promo>
```

```
  </product>
```

```
  <product>
```

```
    <!-- a kind of blue oranges -->
```

```
    <name>Blue Ory</name>
```

```
    <description />
```

```
  </product>
```

```
  <product>
```

```
    <name> having taste of  </name>
```

```
  </product>
```

```
</products>
```

semi-structured data



flexibility

a possible product catalog used by an e-shop

XML – Attributes

Attribute

describes a specific property (characteristic)
about a particular occurrence of an element

XML – Attributes

Attributes are only specified inside a start tag

```
<anthology status="draft" date="2025-12-15">  
  ...  
</anthology>
```

```
<student xml:id="0314159265" account="Tu.Pi">  
  <name initial="I">Tuxy Pinguinnescool</name>  
</student>
```

XML – Attributes

Attributes can be defined in any order

```
<Button id="@+id/sync_settings" text="@android:string/ok" />
```

≡

```
<Button text="@android:string/ok" id="@+id/sync_settings" />
```

real examples: Android applications

android.googlesource.com/platform/packages/apps/

XML – Attributes

Attribute names are case sensitive

```

```

≠

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

XML – Attributes



Attribute value is **mandatory** to be delimited
by quotes or apostrophes

attributes with no value are not accepted

XML – Attributes

```
<form action=process.php method="GET" >  
  <label for=search">Search:</label>  
<input default type=search placeholder= /></form>
```

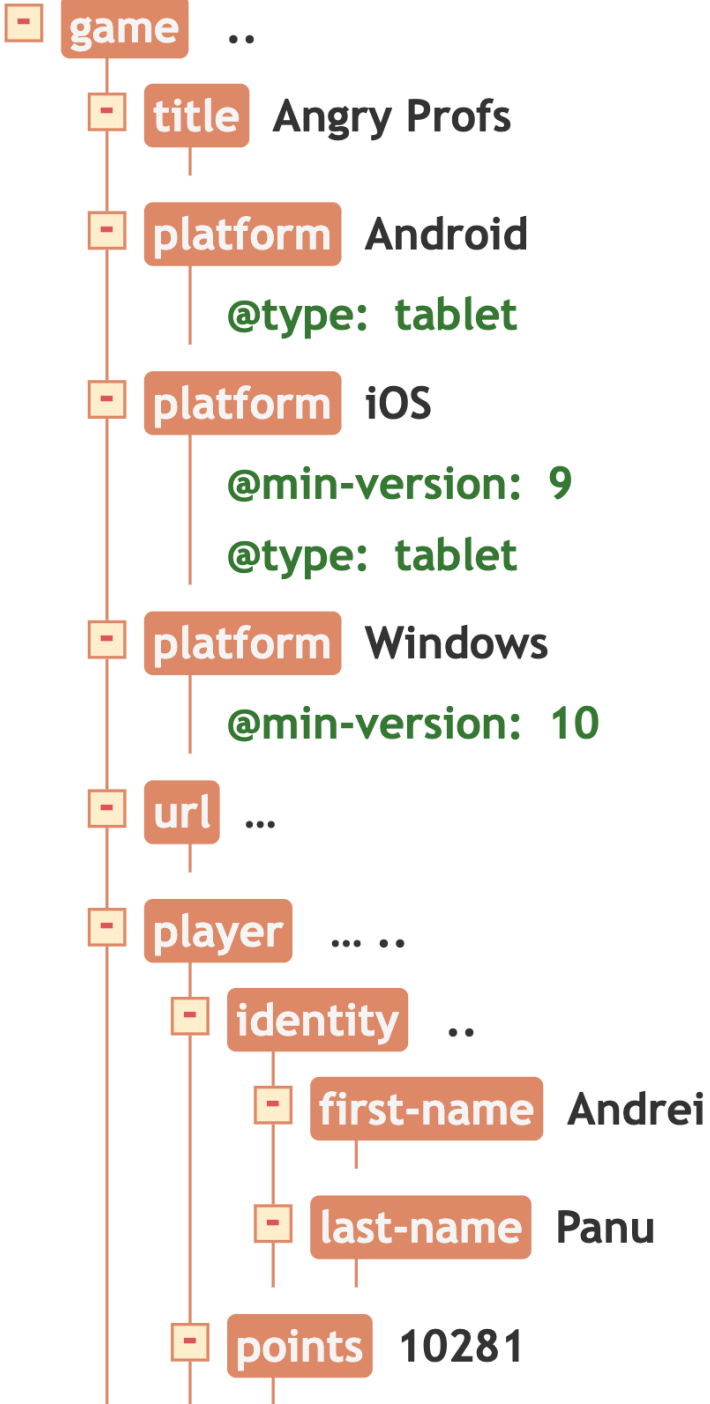
incorrect!

```
:</label> Expected attribute's quoted string value.  
type=search placeholder= /></form>
```

```
<game>
  <title>Angry Profs</title>
  <platform type="tablet">Android</platform>
  <platform min-version="9" type="tablet">iOS</platform>
  <platform min-version="10">Windows</platform>
  <url>...</url>
  <player>
    <identity>
      <first-name>Andrei</first-name>
      <last-name>Panu</last-name>
      <!-- possibly, other info.. -->
    </identity>
    <points>10281</points>
    ...
  </player>
</game>
```

XML data regarding
an electronic game

visualizing the hierarchical structure of XML data



XML – Entity References

Goal:

coding and referring a part of a document

syntax:

`&identifier;`

or

`&#number;`

XML – Entity References

advanced

Default entities – similar to HTML ones:

< (<) **>** (>) **&** (&) **"** (")

Character entity references:

** ** (non-breaking space – ** ** for HTML)

ă (“ä” – ISO-8859-2 and Unicode character sets)

❀ (“☼” symbol – Unicode)

XML – Sections

Certain parts of a document
could require a special treatment

CDATA – excludes the XML processing

XML – Sections

```
<script type="application/javascript">
```

```
if (visits < 10) { // not a regular visitor  
    $("#message").html("<p>Hi!</p>");  
}
```

```
</script>
```

```
if (visits < 10) { // not a regular visitor  
-----^
```

XML Parsing Error: not well-formed
Line Number 3, Column 13

XML – Sections

advanced

```
<script type="application/javascript">
/*<![CDATA[*/
if (visits < 10) { // not a regular visitor
    $("#message").html("<p>Hi!</p>");
}
/*]]>*/
</script>
```

the XML processor will not interpret the syntax of JS code

XML – Processing Instructions

advanced

Include information regarding (external) applications to be invoked for content processing

<?processing-instruction ... ?>

XML – Processing Instructions

advanced

```
<script>
```

```
<?php
```

```
    echo "<p>Hello!\n</p>";
```

```
?>
```

```
</script>
```

the XML processor might invoke, on the server,
the PHP engine in order to run the program

XML – Space Processing

White spaces – e.g., space, TAB, NL (New Line) or CR (Carriage Return) characters – have no significance

```
<VisualAsset id="obsObject">  
  <enabled>true</enabled>  
  <zOrder>0</zOrder>  
  <Orientation>  
    <roll>90</roll>  
    <tilt>90</tilt>  
    <heading>90</heading>  
  </Orientation>  
</VisualAsset>
```

≡

```
<VisualAsset id="obsObject">  
  <enabled>true</enabled><zOrder>0  
  </zOrder><Orientation><roll>90</roll>  
  <tilt>90</tilt><heading>90</heading>  
  </Orientation></VisualAsset>
```

ARML (Augmented Reality Markup Language) markups

XML – Overview

an XML document is composed of node types:

elements

attributes

comments

processing instructions

document type definition (DTD)

XML – Family

advanced

XML (Extensible Markup Language)

syntax

XML Information Set – Infoset

(abstract) data model

XLL (Extensible Linking Language)

XLink – links between documents

XPointer – resource relative locators

XSL (Extensible Stylesheet Language)

transforming & formatting: XSLT + XSL-FO

XQuery (with XPath)

XML data querying

XML – Usages

Content structuring/formatting
(data presentation formats)



in the Web browser:

(X)HTML (Extensible HTML), HTML5

www.w3.org/TR/html/



vector graphics:

SVG (Scalable Vector Graphics)

www.w3.org/Graphics/SVG

XML – Usages

Representation of various types of content



mathematical expressions: MathML

www.w3.org/Math/



synchronized multimedia data:

SMIL (Synchronized Multimedia Integration Language)

www.w3.org/TR/SMIL/

XML – Usages

Representation of various types of content



documentations:
DocBook (Documentation Book)
docbook.org

info processed by office applications – e.g., Open Office:
ODF (Open Document Format)
docs.oasis-open.org/office/OpenDocument/v1.3/os/part1-introduction/OpenDocument-v1.3-os-part1-introduction.html

XML – Usages

Representation of various types of content



Web syndication – newsfeeds:

RSS (Really Simple Syndication)

www.rssboard.org/rss-specification

Atom Syndication Format

datatracker.ietf.org/doc/html/rfc4287




electronic publications (e-books): EPUB

github.com/w3c/epub-specs/

XML – Usages

Representation of various types of content

 medical information (EHR – Electronic Health Records)

HL7: www.hl7.org/implement/standards/

 electronic businesses

FpML–Financial products Markup Language: www.fpml.org

 governmental information

NIEM–National Information Exchange Model: niem.github.io

XML – Usages (other domains)

BeerXML

BDML (Biological Dynamics Markup Language)

CAP (Common Alerting Protocol)

CML (Chemical Markup Language)

COLLADA (COLLABorative Design Activity)

DFXML (Digital Forensics XML)

GPX (GPS Exchange Format)

MEI (Music Encoding Initiative)

RTML (Remote Telescope Markup Language)

SSML (Speech Synthesis Markup Language)

STAR (Standards for Technology in Automotive Retail)

TEI (Text Encoding Initiative)

What if we choose names of markups/attributes already defined by other XML-based languages?

```
<event uri="https://stagiipebune.ro/">  
  <name xml:lang="ro">Stagii pe Bune</name>  
  <year>2026</year>  
</event>
```

```
<participant>  
  <name uri="mailto:tux@info.uaic.ro">  
    Tuxy Pinguinnesscool  
  </name>  
  <year kind="Bachelor">2</year>  
</participant>
```

```
<event uri="https://stagiipebune.ro/">  
  <name xml:lang="ro">Stagii pe Bune</name>  
  <year>2026</year>  
</event>
```

conflict!

?

```
<participant>  
  <name uri="mailto:tux@info.uaic.ro">  
    Tuxy Pinguinnesscool  
  </name>  
  <year kind="Bachelor">2</year>  
</participant>
```

<name> event name \neq **<name>** person name
<year> calendaristic year \neq **<year>** study year

XML – Namespaces

advanced

XML namespace

denotes a vocabulary used to qualify – in a unique way – the XML elements/attributes

XML – Namespaces

The defined vocabulary – a collection of element and attribute names, plus their structure – could be denoted by an URI

XML – Namespaces

The defined vocabulary could be denoted by a URI

xmlns attribute specifies that URI;
optionally, assigns a unique identifier
to each used vocabulary

W3C specification:

www.w3.org/TR/xml-names/

```
<c:calendars xmlns:c="http://www.calendar.info">
  <e:participant xmlns:s="http://www.info.uaic.ro/Students/"
    xmlns:e="http://www.info.uaic.ro/Events/">
    <s:name>Tuxy Pinguinnescool</s:name>
    <s:year s:kind="Bachelor">2</s:year>
    <c:calendar>
      <e:event xml:id="SpB">
        <e:name xml:lang="ro">Stagii pe Bune</e:name>
        <e:year>2025</e:year>
      </e:event>
      <e:event xml:id="GSoC">
        <e:name xml:lang="en">Summer of Code</e:name>
      </c:calendar>
    </e:participant>
  </c:calendars>
```

no conflicts!

How can we process XML documents?

XML – Processing

Types of XML processing

manual processing

e.g., regular expressions 🥲

XML – Processing

Types of XML processing

object-oriented processing

DOM (Document Object Model)

non-DOM

XML – Processing

Types of XML processing

event-driven processing

SAX (Simple API for XML)

XPP (XML Pull Parsing)

XML – Processing

Types of XML processing

simplified processing

Simple XML

XML – Processing

Types of XML processing

specific processing

via specialized APIs in order to process
particular document types
e.g., KML, RSS, MathML, SVG,...

Live examples using SimpleXML in PHP

Summary