Script language: Python

xml.dom.minidom

Cédric Saule

Technische Fakultät
Universität Bielefeld

13. Februar 2016

XML - eXtensible Markup Language

- Data are structured (texts, images, measurement results).
- A mechanical processing requires the knowledge of the structure.
- Automated processing requires uniform data format.

Wanted: formalism to describe any structures.

Solution: XML

- XML structure textual data.
- XML is a meta-language for defining markup languages.
  - Which elements / attributes are present.
  - How to nest them ?

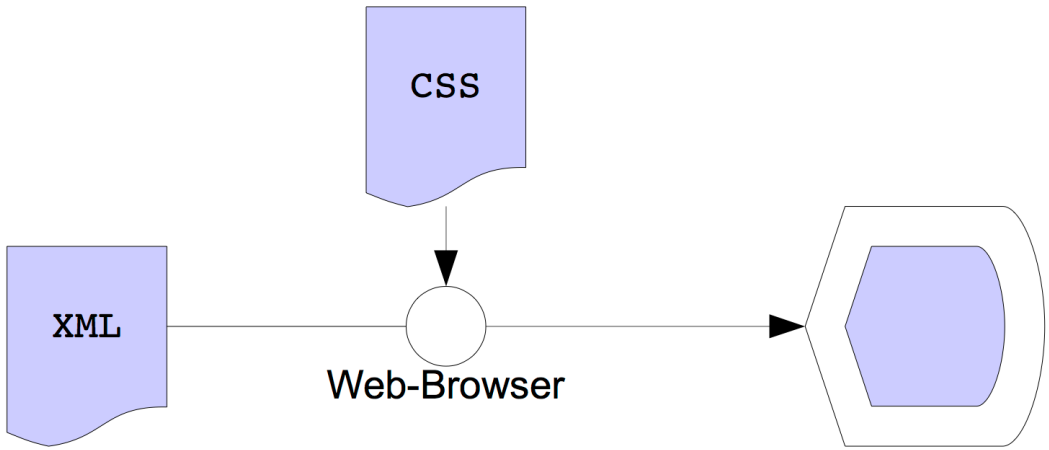
XML language examples

- XHTML 1.1
- SVG
- RSS 2.0
- RDF
- MathML

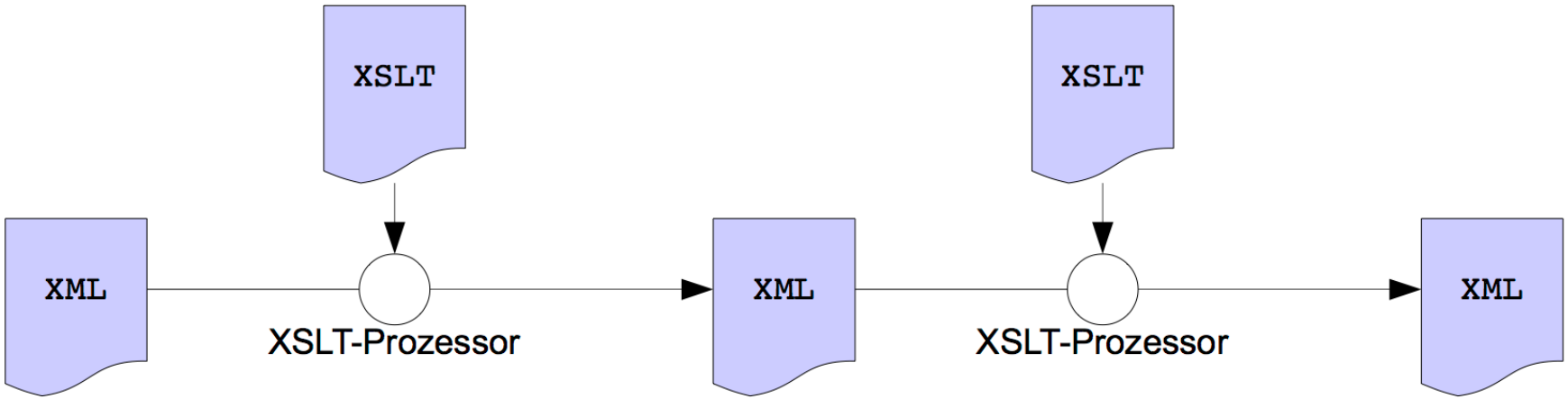
```
<?xml version="1.0" encoding="UTF-8"?>
<Root element>
  <Element>...</Element>
  <Element>
    <Standalone element Attribute="" />
  </Element>
  <Standalone-Element />
</Root element>
```

XML - So what?

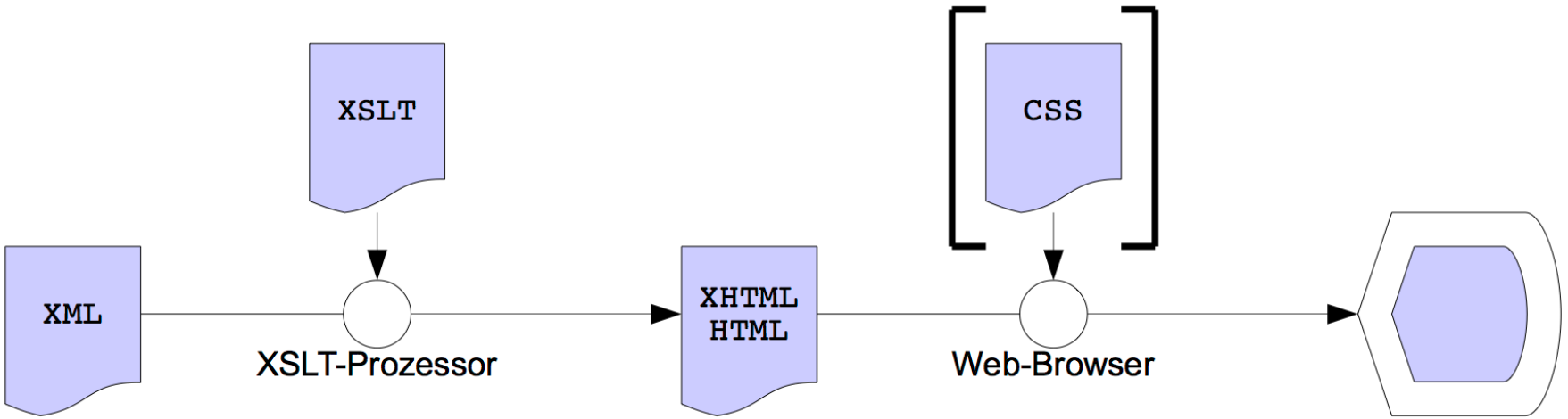
- XML is purely used for data representation.
- Representation of secondary problem.
- XML must be further processed.



XML transformation



XML transform & representation



An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

XML declaration

Optional encoding:
encoding="UTF-8"
encoding="ISO-8859-1"

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
     or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Root

Documents XML obviously use a root element.

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Element

We can follow from the root of any number of child elements. They are also called node elements.

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Opening and closing tags.

An element obviously uses an opening and a closing tag:
<elem> – Opening
</elem> – Closing

Alternatively, a tag can also be empty: <elem/>

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Text node

Elements can contain some texts as content, so they have a text node as a child node.

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Entities

Special characters are set as entities – Text macro.

They are used to encode special characters, reserved characters and strings.

In addition, it can be defined again. The winner: DTD (Document Type Definition).

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp;amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Attributes

Elements can have attributes to store additional informations. The key/value pair only occurs in the opening tag and must be named only once.

An example

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
    or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
</slide>
  <slide>...</slide>
</presentation>
```

Nested Elements

In text nodes, embedded elements can occur; We have to take care to nest them in the right way:

<a>...
vs.
<a>...

Special characters

- Writing in capital letters or not is relevant:
  <elem>Does not work</Elem>
- Reserved special characters:

  < – < > – > & – &
  ' – ' " – "

- Multiline comments: <!−−*COMMENTARY* −−>
  - Can not occurs: −−
  - No possible nesting

Well formedness and validation

If all the requirements have been met, the XML document is well formed and can be correctly processed, when a DTD / schema is specified.

Check your validation:

Command line xmlwf, xmllint

Online tools W3C, Validome

Exercise – Media database

Creates a XML document that contains a media database to represent musics. Check the well-formedness using the command-line tools or online tools.

Insert an example of three entries (including empty elements in an entry).

Namespaces

- Allow the use of different XML languages in a document.
- Similar to a module (Python, Java) or a telephone preselection $\rightarrow$ distinction between similar elements.
- Are represented by the URI (Uniform Resource Identifier)
  - Must not represent a valid URL (Uniform Resource Locator)
  - If there is a valide URL, it should be found:
    - DTD (Document type definition).
    - Schema XML.

Namespaces

```
<html xmlns="http://www.w3.org/1999/xhtml">
  ... XHTML-Elements
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    ... MathML-Elements
  </math>
  ... XHTML-Elements
</html>
```

Attribut xmlns:pf="http://..." bind the namespace pf on the URI http://...

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
  ... XHTML-Elements
  <m:math>
    <span>Here is XHTML. Variable x: <m:i>x</m:i></span>
  </m:math>
  ... XHTML-Elements
</html>
```

XPath & DOM representation

There are different technics to navigate through an XML document.

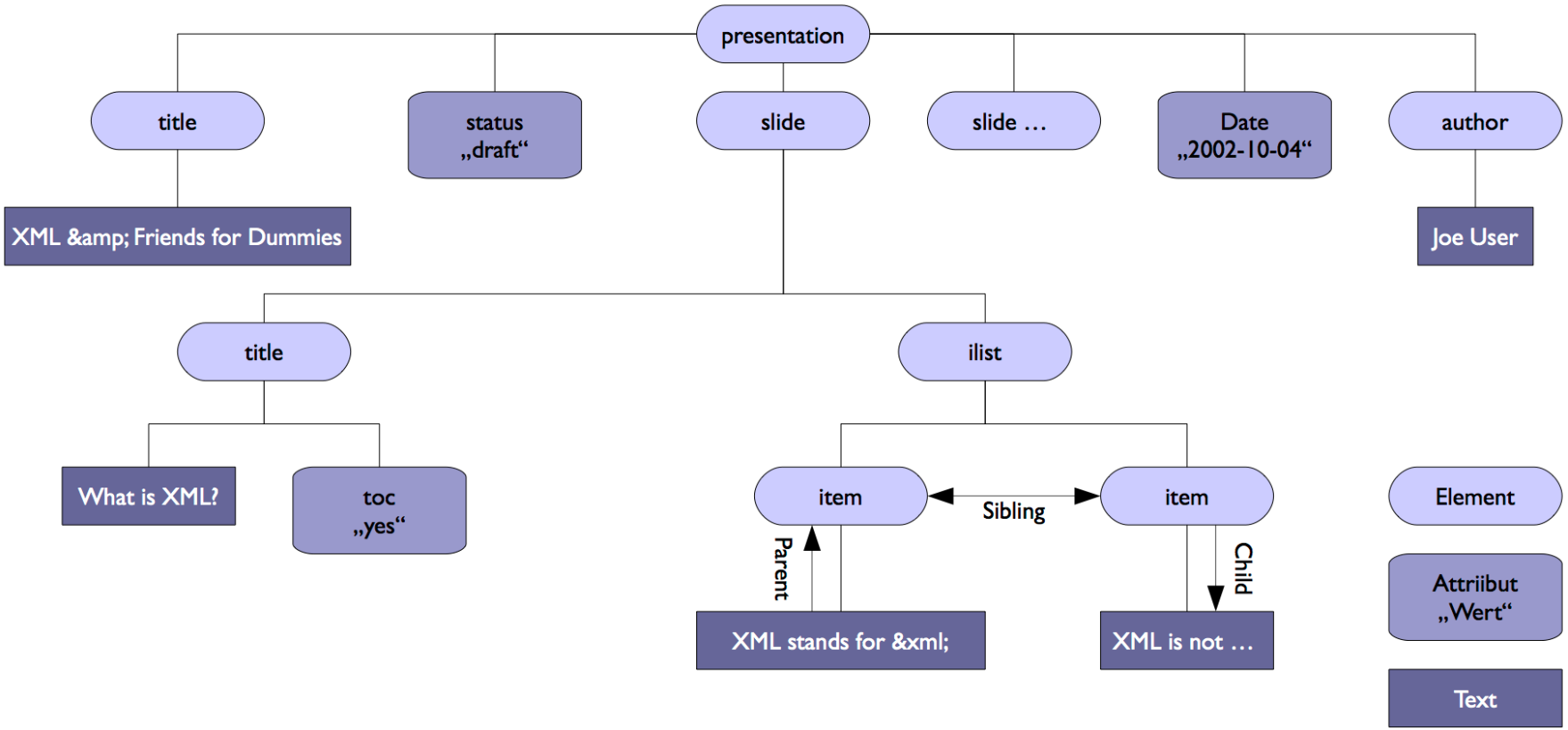
XPath Navigation.

DOM representation Navigation and Modification.

Both processes see the document as a tree, with:

- Nodes: Element, Attribut, Text, comment, Namespace, PI(Processing Instruction).
- Navigation along an axis.
- Test optional nodes (Conditions).

Tree representation



XPath

- Starts the search from the current context node.
- Find a destination node during localization:
  Axis :: NodeTest[Predicate1][Predicate2]

Axis	Path separated with / or axis specification.
TestNodes	$*$ all kinds of nodes, NAME a node or a function call Text() or comment()
Predicate	Index, Relations characters, String functions, Operators, Function of nodes merge.

XPath – Example

```
/* – Root element
//Interprete – All elements on the interpreted axis
//Title/Producers[2] – The two produce a title
//Title/Producers[2][@gema='ja'] – The two produce
a title which is member of GEMA.

attribute::* – All attributes of the
        context's nodes
child::text() – All texts nodes of the
context's nodes

. – The current context node
```

Representation DOM

- Document Object Model (DOM) specification for accessing XML documents.
- Implementations available in many languages: Javascript, JDOM Java, LibXML each C, Python, Perl ...
- By specification: Uniform functions, function names, function.
- Read and write access.

Working with DOM

- Document XML is read and parsed – Parser
- A XML document is available as a tree structure, with the following node types:

Document	Node	Element
Attr	Text	Comment
CDATA	Namespace	DTD
Processing Instructions		

- Variables represent nodes of a certain type $\rightarrow$ Functions on variables are context-dependent.
- Nodes can use an unique ID code name (with / without namespace), names, family relationships and are driven by means of XPath expressions.

The modul xml.dom.minidom is a special lightweight DOM implementation for Python.

- Implements DOM 1.0 completely.
- DOM 2 feature list only partially, but enough for us.
- Full OO approach.

Parsing a document

Import module `import xml.dom.minidom as dom`

Data parsing `dom.parse(<FILENAME>)`

String parsing `dom.parseString(<STRING>)`

Empty DOM tree `dom.Document()`

DOM tree serialization

as string	toxml([ENCODING]), Default-Encoding: UTF-8
in Data	writexml(WRTR[, indent[, addindent [, nl]]]) , with:
WRTR	Often file Instance.
indent	Insertion of the start node.
addindent	Insertion of all further child nodes.
nl	New line Character.

```
> f = open("stuff.xml", "w")
> domTree.writexml(f, "", "\t", "\n")
> f.close()
```

Create items

- Element nodes

```
element = doc.createElement(qname)
element = doc.createElementNS(nsURI, qname)
```

- Text nodes

  text = doc.createTextNode(data)

- Commentaries nodes

  comment = doc.createComment(data)

- Attributes nodes

  attribute = doc.createAttribute(nsURI, qname)

Attributes nodes

Creating an attribute nodes, so what ? What is the value ?

```
> aNode.value = <StrAsValue>
> node.setAttributeNode(aNode)
```

Or just create the node:

```
node.setAttribute(<NAME>, <StrAsValue>)
```

And again, get rid of it...

```
node.removeAttribute(name)
node.removeAttributeNode(aNode)
```

Finding list of nodes

- On the element id

```
node = doc.getElementById(id)
```

Warning! The attributes must be of type id!

- On the element name

```
nodes = doc.getElementsByTagName(tagname)
nodes = doc.getElementsByTagNameNS(nsURI, tagname)
```

Replace attachments, deleting nodes

```
child = node.appendChild(children_node)

node.replaceChild(new_node, old_node)

child = node.removeChild(child_node)
```

Exercise – minidom is pretty big

Expand the XML document created in the previous exercise to another entry with the help of minidom in Python.

Read here for your document, find the node to be expanded, add new nodes, cleave the DOM tree at the corresponding place and save the XML document under a new name.