PB138 — Concepts, XML Document Structure

Outline

- Logical and physical structure
- Concepts: *node, element, attribute, processing instruction, text node, comment*

XML document structure

Fundamental requirement to all XML doc: it must be **well-formed**:

- It contains prolog (heading) andexactly one root element. Before and after the root element, there can be processing instructions, comments (Misc).
- It meets all the well-formedness constraints given in the specification.
- Each of the parsed entities which is referenced directly or indirectly within the document is well-formed.

Higher-level requirement for an XML document: it can be *valid*.

XML document structure — further info

- Look at tutorial on XML Fundamentals in English
- ToC for XML

XML document structure

- XML document structure we distinguish between:
  - **physical** and
  - **logical** structure.
- Application programmers are usually interested just on the *logical structure*,
- while for the authors of content, XML editors, processors may also the *physical structure* be important.

Physical and logical structure

Logical structure

A document is divided into elements (one of them is the root), their attributes, text nodes in elements, processing instructions, notations, comments.

Physical structure

One logical doc may be stored in one or more entities; at least in the document entity.

Composition of logical structure

- node which is a generic type, we further distinguish:
- **element** (somethimes incorrectly called "tag" — while tag is just the opening and closing markup, not the whole element)
- **attribute** (always attached to some element)
- **text node** (the text between some markup)
- **processing instructions** (not containing text content nor attributes, just for processing purposes)
- **comments** (usually targeted to human readers)

Prvky logické struktury (česky)

- uzel (element, atribut, textový uzel, instrukce pro zpracování, komentář)
- element
- atribut
- textový uzel
- instrukce pro zpracování
- komentář

Elements

are objects delimited by start- and end-tags, examples:

```
<body background="yellow">
   <hl>text node – content of element hl</hl>
   <p>text node – content of element p</p>
</body>
```

Elements — empty

If an element is empty (no child elements, neither text content inside), then we write just empty element tag, eg.:

```
<tagname tagattributel tagattribute2 ... />
```

such as

```
<hr width='507'/>
```

or equivalently (from logical viewpoint):

```
<hr width='507'></hr>
```

Attributes

- Always placed in the start-tag of an element, eg. `<hr width='507'>`
- The **physical order** of attributes within a start-tag is **NOT significant** and generally is NOT considered.
- Attributes are thus simply "attached" to elements, carry "additional info" to elements - eg. its `ID`, required formatting (style) in case of (X)HTML, or links to other elements

Attributes vs. elements

- Conceptually, we could replace all attributes with elements but we keep attributes to maintain readability.
- The attribute content should **NOT be further structured**
- Attribute value is not structured at least according to XML standards. An application may see it other way but generally it is not recommended, cf. the same holds for attributes in relational data model.

How to write attributes

- An attribute is composed of its **name** and **value**.
- Attributes are inserted in the start-tag which may be empty.
- Attribute value is always in quotes (`'`) or doublequotes (`"`) add separated by a `=` from the attribute name.
- Writing `width="750"` or `width = "750"` or `width='750'` means completely the same.
- For attribute names the same rules as for element names hold.
- In one element, there can never be *two or more attributes with same name*.
- If namespaces are used, neither two attributes with same name belonging to the same namespace are allowed.

Attributes

```
<table border='l'>
   <tr>
    <td>jedna</td>
    <td>dve</td>
   </tr>
   <tr>
    <td>tri</td>
    <td>ctyri</td>
   </tr>
</table>
```

Text node

- They carry textual information, textual content.
- Eg. in the next sample, the text `ahoj!` is the text node and not the whole element `em`

```
<em>ahoj!</em>
```

Processing instructions

- Processing-instructions are written using `<?target content?>` markup.
- They inform an application about the expected processing or setting.
- They do not carry content.

```
<?xsl-stylesheet href="mystyle.xsl"?>
```

- `href` does not mean an attribute; Processing-instructions do not contain attributes.

Notations

- Notation is enclosed in `<!NOTATION name declaration>`
- It is mostly used to describe binary / non-XML entities - eg. images GIF, PNG,...
- It is a declaration how to process the binary data.

Comments

- Similarly to HTML - comment is enclosed into `<!--content-->`

- The comment content is content, NOT the the whole comment including markup.
- Comments are usually not important for processing but it may depend on application, eg. Servlet-side Includes (SSI) use comments.
- Parsers therefore should be able to forward comments to the applications.
- SAX parsers ignore this in version 1; do so in version SAX2, in Java the package org.xml.sax.ext.

Entity

- **Entity** is a basic unit of physical document composition.
- Corresponds to a *character*, *string*, or whole *file*.
- Parsers should process the entities so that the applications do not know about them.

Document node

We distinguish:

Document node
: parent of the root element; may contain also Pis, notations, DOCTYPE etc. and

Root element
: is the core part of an XML doc. In every file, there is just one.

In more details...

in the next chapter XML family standards.