XML Example

```
<?xml version='1.0' encoding='UTF-8' ?>
<!-- first XML example -->
<message id="123456">
  <text>Hello World!</text>
</message>
```

- **version**: The version number of the XML document.

- **encoding**: Character encoding - 'ISO-8859-2' refers to Latin-2 encoding.

- **comments**: Comments can be placed between <!– –>, and they can span multiple lines.

- **tagging data**: Data is labeled using **tags**. For example, <name></name>, where <name> is the opening tag and </name> is the closing tag. If the tag doesn't contain data, the closing can be simplified like <name />.

- In this example, “Hello World!” is stored in the <text> tag.

- An XML file contains a **root element** (or **root tag**) to which all other elements belong, in this case <message>. Elements that contain other elements are called **parent elements**, and the elements within them are called **child elements**.

- Elements can contain an unlimited number of **attributes** (properties), such as the id attribute = “123456”. Attribute names are not limited in length, but every attribute name must start with a letter.

Special Characters in XML

In XML, special characters are represented as follows:

- & - &

- < - <

- > - >

- ''' - '

- “ - "

CDATA Section

A **CDATA** section can contain embedded data, such as the following example:

```
<script language="JavaScript" type="text/javascript">
  <![CDATA[
    function sayHello() {
      document.write("Hello World!");
    }
  ]]>
</script>
```

XML Namespaces

Since creators of XML documents use their own vocabulary to build XML, **name conflicts** are possible. For example, the `<student>` tag might be too generic. Using **namespace prefixes**, the element can be specialized:

```
<miskolc:student>John Smith</miskolc:student>
```

Here, the qualified element is `student` and the namespace prefix is `miskolc`.

```
<?xml version="1.0" encoding="UTF-8"?>
<data xmlns:unimiskolc="www.uni-miskolc.hu">
  <unimiskolc:file filename="aula.jpg">
    <unimiskolc:description>Photo of the university
hall</unimiskolc:description>
    <unimiskolc:size width="200" height="100" />
  </unimiskolc:file>
</adatok>
```

In this example, the namespace prefix `unimiskolc` is defined to avoid name conflicts, and it links the data to the **www.uni-miskolc.hu** namespace.

XML Interpreters

XML interpreters process XML documents by performing **syntactic validation**.

Definition

An XML document is considered **“well-formed”** if it is **syntactically flawless**.

This means: - It has a **single root element**.

- Every opening tag has a corresponding closing tag.

- The attributes are properly specified.

Therefore, the term **“well-formed”** does not refer to the visual formatting of the XML file with tabs, spaces, or its human-readable structure!

Example

```
<root><element>abc</element><a>this worng<b></root>
```

This XML is **syntactically incorrect** because the `<a>` tag is not closed.

Example

The following example is neatly formatted but still **not well-formed**:

```
<CONTACT>
   <NAME>Frank Lee</NAME>
   <EMAIL>flee@flee.com
</CONTACT>
```

Here’s the correct version:

```
<CONTACT>
   <NAME>Frank Lee</NAME>
   <EMAIL>flee@flee.com</EMAIL>
</CONTACT>
```

What’s the difference? In the incorrect version, the `<EMAIL>` tag is not closed, causing the document to be invalid, even though it is visually well-structured.

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:techcomm:xml_2

Last update: **2024/10/07 17:30**