Chapter I

XML

XML - Literature

- Young, Michael J. (2002)
  - XML. Schritt für Schritt. (Microsoft Press)
- Bongers, Frank (2004)
  - XSLT – Das umfassende Handbuch (Galileo Press)
- Van der Vlist, Eric (2003)
  - XML Schema (O'Reilly)

What is XML?

- XML (extensible markup language) is a highly flexible standard for structured, human & machine readable data streams
- it defines the syntax to create arbitrary yet similar markup languages (the so-called XML-applications)
  - example: XHTML – the XML-compatible version of HTML
- the basic idea is the strict separation between data and their representation
- it is a derived subset of SGML

Now is XML a markup language?

XML Applications

- an XML application consists of three parts
  - structure
    - defines elements, attributes, etc., which may be used in corresponding XML documents, either in form of a document type definition (DTD) or an XML scheme
  - one or several XML documents
    - contain the data
  - processing directives
    - define how the data is represented, e. g. using cascading style sheets (CSS) or the extensible stylesheet language (XSL) or the document object model (DOM)

- the official w3-specification of XML can be found here:
  - http://www.w3.org/tr/rec-xml

The Purposes of XML

- 1. XML shall be straightforwardly usable over the Internet.
- 2. XML shall support a wide variety of applications.
- 3. XML shall be compatible with SGML.
- 4. It shall be easy to write programs which process XML documents.
- 5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- 6. XML documents should be human-legible and reasonably clear.
- 7. The XML design should be prepared quickly.
- 8. The design of XML shall be formal and concise.
- 9. XML documents shall be easy to create.
- 10. Terseness in XML markup is of minimal importance.
    - source: http://www.w3.org/tr/rec-xml

Well-formed XML Documents

- XML documents are well-formed when having an XML-conform syntax and keeping all relevant rules given in the XML specification
- an offence against any of these rules causes the XML processor to come up with a serious error message
- well-formed XML documents consist of needed and optional parts
  - needed:
    - XML declaration
    - document element (at least a root element)
  - optional:
    - comments
    - empty lines
    - processing directives

Example

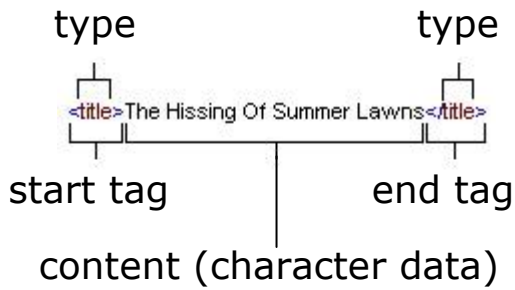
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: books.xml -->

<?xml-stylesheet type="text/css" href="books.css"?>

<all_books>
  <book>
    <title>Harald Butter und der Schrein der Bleichen</title>
    <author>Johanna Krauling</author>
    <price>29,95</price>
  </book>
  <book>
    <title>Harald Butter und der Meuterelch</title>
    <author>Johanna Krauling</author>
    <price>29,95</price>
  </book>
</all_books>

<!-- Comments, empty lines and processing directives may be inserted after the document element too. -->
```

Mistakes

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: books.xml -->

<?xml-stylesheet type="text/css" href="books.css"?>

<book>
    <!-- Find 5 Mistakes! -->
    <title>Harald Butter und der Schrein der Bleichen</Title>
    <author>Johanna Krauling</author>
    <price>29,95</price>
    <xml>document type</xml>
</book>
<book>
    <title><b>Harald Butter und der Meuterelch</title>
    <author>Johanna Krauling</b></author>
    < price>29,95</price>
</book>

<!-- Comments, empty lines and processing directives may be inserted after the document element too. -->
```


mls

Elements


type
type
<title>The Hissing Of Summer Lawns</title>
start tag
end tag
content (character data)

Element Types (I)

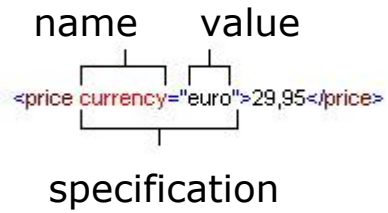
- there are no predefined element types within XML
- invalid names for element types (examples):
  - `<1st_element>…</1st_element>`
    - the first character may not be a number (only characters and underline are allowed)
  - `<element a>…</element a>`
    - space is not allowed
  - `<price_in_pounds/dollars>…</price_in_pounds/dollars>`
    - slash is not allowed
  - `<:result>…</:result>`
    - the first character may not be a double point
  - `<composer>…</Composer>`
    - start- and endtag must be written exactly the same way

Element Types (II)

- instead of `<empty></empty>`, empty elements can be written that way: `<empty/>`
- element types beginning with the prefix "xml" are reserved for further "standardisations"
  - actually, present browsers don't throw an exception, but it is recommended by the w3c not to use names beginning with "xml"
- the content of the element (the data between start- and end-tag) may not contain <, > and & (& introduces an entity reference)

Attributes (I)

- an elements start tag or an empty element may contain one or more attributes



Attributes (II)

- invalid attributes (examples):
  - `<dog source="bobtail_01.dog" source="bobtail_02.dog">`
    - the same attribute may not be used twice within one element
  - `<hitlist 1-10="top_ten.txt">`
    - the first character may not be a number (only characters and underline are allowed)
  - `<table head=""id"">`
    - use `head="'id'"` or `head='"id"'` instead
  - `<album type="<cd>">`
    - the attribute may not contain < (> is allowed within the value, but not within the attributes name)
  - `<weather forecast="cold & cloudy">`
    - & is only allowed when using a character or entity reference
  - `<setup read f i r s t !="readme.txt">`
    - the attributes name may not contain spaces

Attributes (III)

- attribute names beginning with the prefix "xml" are reserved for further "standardisations"
  - actually, present browsers don't throw an exception, but it is recommended by the w3c not to use names beginning with "xml"

<![CDATA[ ]]>

- CDATA sections start with `<![CDATA[` and end with `]]>`
- in between you can enter various characters, including <, >, & or even (X)HTML tags that are not to be interpreted by the XML processor (of course `]]>` is forbidden, as it would end the CDATA section)
- all characters are interpreted as being not markup code
- `<![CDATA[` is an XML keyword and therefore has to be written in capital letters
- CDATA sections may be inserted on every position where character data are allowed

Example

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

<basics>
    <xhtml>XHTML-files are divided into <![CDATA[<html>- & <body>-Tags.]]></xhtml>
</basics>

<!-- This file is well-formed XML. -->
```

Namespaces (I)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: books.xml -->

<collection>
  <item>
    <title>The Adventures Of Huckleberry Finn</title>
    <author>Mark Twain</author>
  </item>
  <item>
    <title>Madrapour</title>
    <author>Robert Merle</author>
  </item>
</collection>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: cds.xml -->

<collection>
  <item>
    <title>Selling England By The Pound</title>
    <interpret>Genesis</interpret>
  </item>
  <item>
    <title>Tales From Topographic Oceans</title>
    <interpret>Yes</interpret>
  </item>
</collection>
```

Namespaces (II)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: library.xml -->

<collection
   xmlns:book="http://myhomepage.com/books"
   xmlns:cd="http://myhomepage.com/cds">

   <book:item>
      <book:title>The Adventures Of Huckleberry Finn</book:title>
      <book:author>Mark Twain</book:author>
   </book:item>
   <book:item>
      <book:title>Madrapour</book:title>
      <book:author>Robert Merle</book:author>
   </book:item>
   <cd:item>
      <cd:title>Selling England By The Pound</cd:title>
      <cd:interpret>Genesis</cd:interpret>
   </cd:item>
   <cd:item>
      <cd:title>Tales From Topographic Oceans</cd:title>
      <cd:interpret>Yes</cd:interpret>
   </cd:item>
</collection>
```

Namespaces (III)

- a namespace is declared as a special kind of attribute within the start tag of a specific element
- it separates elements with identical names but different meaning by referring to a uniform resource identifier (URI)* ** ***
- neither the XML application nor the XML processor is accessing that URI!
- an element may have two attributes with the same name – they only have to be separated by at least one namespace
- even a standard namespace can be declared, which is then valid for all undeclared elements

* URIs contain all uniform resource locators (URLs) and all uniform resource names (URNs)

** details: http://www.w3.org/adressing/

*** it *should* be a URI – it *can* be anything else (e.g. a number)

Standard Namespaces

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- filename: library.xml -->

<collection
   xmlns="http://myhomepage.com/books"
   xmlns:cd="http://myhomepage.com/cds"
   xmlns:remaster="remaster">

   <item>
      <title>The Adventures Of Huckleberry Finn</title>
      <author>Mark Twain</author>
   </item>
   <item>
      <title>Madrapour</title>
      <author>Robert Merle</author>
   </item>
   <cd:item>
      <cd:title>Selling England By The Pound</cd:title>
      <cd:interpret>Genesis</cd:interpret>
   </cd:item>
   <cd:item year="1973" remaster:year="2002">
      <cd:title>Tales From Topographic Oceans</cd:title>
      <cd:interpret>Yes</cd:interpret>
   </cd:item>
</collection>

<!-- Using the Attribute xmlns="" (an empty namespace) you can keep elements out of a standard namespace. -->
```

Testing your Skills

- Create a well-formed XML-document in iso-8859-1 containing two or three of your favourite music albums. Each album consists of its name, the artist's name and the year it was published. Comment each album with a five star rating and display it with a fictional CSS file. Add an information about its type to each album (e.g. cd, mc or lp), which must not be displayed in the browsers window. All elements within the root element (except the year) shall become part of a standard namespace.