Data Storage and Structure

Responsible persons:

Regula Stopper

(Overall)

Andreas Neumann

(Content)

Samuel Wiesmann

(Revision)

Olaf Schnabel

(Revision)

Table Of Content

1. Data Storage and Structure

Learning Objectives

You will be able...

- ...to distinguish four data storage types and choose the right storage type for a specific project.
- ...to explain what XML is and what it is used for.
- ...to list the main properties of XML.

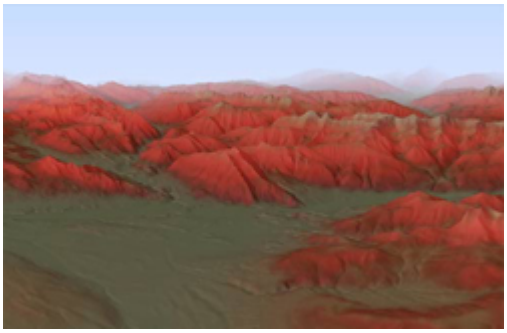
Introduction

Imagine producing a multimedia application with the title "The Red Deer in Switzerland" and you possess the following data (pictures, text and data table):

Picture of Red Deer (Atlas of Switzerland 2.0 2004)

Red deer, often called king of the forest, reaches a head-body length of approx. 2 m, a shoulder height of up to 120 cm in females and 150 cm in males, as well as a weight of 90 to 120 kg, respectively. It is the largest mammal in Switzerland. Only the male (stag or bull) carries antlers which are shed and regrown each year...

Describing Text (Atlas of Switzerland 2.0 2004)



Picture of occurence of Red Deer (Atlas of Switzerland 2.0 2004)

X-Coord	Y-Coord	Z-Coord	#Red Deer
600050	200040	800	3
600060	200050	768	2
600070	200060	900	4
...	...	...	...

Table with Red Deer Occurence

Before being able to start with the production of a multimedia application, the available data has to be structured and stored. Several possibilities exist to store data. We introduce you four of these possibilities in this lesson. Depending on the given data the storage method has to be chosen and applied. After this lesson, you should be able to define the best storage method for the data examples above.
A file type that is often used to store data is XML, a W3C recommendation. The main purpose of XML is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. Currently, the interoperability of different datasets is considered as one of the major technological challenges within the research community. Because XML helps to solve this problem, we give you a detailed introduction to XML in this lesson.
We will show you the main properties of XML, its advantages and disadvantages. You will get familiar with the XML's syntax and the structure description of an XML document.

1.1. Data Storage

Learning Objectives

You will be able...

- ...to distinguish four storage methods that can be used for data.
- ...to list the advantages and disadvantages for each file type.

Introduction

When producing a multimedia map you have data (geometric and thematic data) that has to be stored and structured in an optimal way. There exist several possibilities how to store the needed data for a project. You can store it in separate files broken down by topic or just store everything in one database, etc. If you store the data in separate files they do not even have to be of the same file types. You are as well able to store parts of the data in text files and parts of it in databases.
In this unit we will introduce you some possibilities how data - thematic or geometric - can be stored and what advantage and disadvantage feature these formats.

1.1.1. Plain Text Files

Plain text files (or simply text files) are files which contain ordinary readable characters such as letters, digits, and punctuation (including space). They also include tabs, return commands, etc. as control characters.
Generally, a plain text file contains characters in an *ASCII* [1]-based encoding without any embedded information such as font information, *hyperlinks* [2] or inline images. (Wikipedia)
Although text files are often meant for humans to read, they are also commonly used for data storage.
The storage of data in plain text files is defined by the following points:

- typically one row per record
- fixed width or delimited
- special character as delimiter such as tab, ";", etc.
- various encodings (ASCII, *ISO* [3]8859-1,UTF8, UTF16)

Keyboard Keys (Wikipedia)

Example of a Plain Text File
Here is a very small example how a structured text might look like.

[1] ASCII (American Standard Code for Information Interchange) is a character encoding based on the English alphabet. ASCII represents text in computers, communications equipment, and other devices that work with text.

[2] An element in an electronic document that links to another place in the same document or to an entirely different document. Typically, you click on the hyperlink to follow the link. It is most commonly used in the World Wide Web to link various documents (Web Pages, pdf-files, etc.).

[3] The International Organization for Standardization is the international organisation responsible for developing and maintaining technical standards.

```
"station";"temperature";"air_pressure";"wind_direction";"storm_force"
"Jungfraujoch";-5;950;270;43.7
"Säntis";2;978;250;25.5
"Rigi";4;1008;15.3;
```

Example of Plain Text File

We use quotes to indicate text and semicolons as a delimiter. The first line is just a header line, not real data.

Advantages
The advantages of plain text files are:

- Compactness of the files,
- Simplicity of the files,
- They can be interpreted by humans.

Disadvantages
The disadvantages of plain text files are:

- not useful for larger datasets, since currently no efficient management tools for larger datasets exist.
- access to individual records and attributes complicated, since the file is only structured with columns and rows.

Usage Scenarios
Plain text files are mainly used for small unstructured datasets, for the import/export to/from spreadsheet programs or for the import into GIS or Mapping Software.

1.1.2. Binary Files

"*A binary file is a file whose content must be interpreted by a program or a hardware processor that understands in advance exactly how it is formatted. [...] A program has to know exactly how the data inside the file is laid out to make use of the file.*" (Whatis.com)

Example of a Binary File
When opening a binary file in a texteditor, it looks like this:

```
00000270h: 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 06 ; ................
00000280h: 00 00 00 69 66 50 68 79 73 41 64 64 72 65 73 73 ; ...ifPhysAddress
00000290h: 00 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 ; ................
000002a0h: 07 00 00 00 69 66 41 64 6D 69 6E 53 74 61 74 75 ; ....ifAdminStatu
000002b0h: 73 00 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 ; s...............
000002c0h: 00 08 00 00 00 69 66 4F 70 65 72 53 74 61 74 75 ; .....ifOperStatu
000002d0h: 73 00 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 ; s...............
000002e0h: 00 09 00 00 00 69 66 4C 61 73 74 43 68 61 6E 67 ; .....ifLastChang
000002f0h: 65 00 00 00 00 00 00 00 00 0A 00 00 00 00 00 00 ; e...............
00000300h: 00 0A 00 00 00 69 66 49 6E 4F 63 74 65 74 73 00 ; .....ifInOctets.
00000310h: 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 0B ; ................
00000320h: 00 00 00 69 66 49 6E 55 63 61 73 74 50 6B 74 73 ; ...ifInUcastPkts
```

Example of a Binary File

Binary is nothing more than a numeral system. Binary files are usually thought of as being a sequence of *bytes* [4], which means the binary digits (bits) are grouped in eights (00000001, 00000011, etc.). (Wikipedia)
You may have recognized that the example above does not include the mentioned sequences of bytes. When opening a binary file in a texteditor such as UltraEdit, it is coded using *hexadecimal* [5] values. The following table shows the coherence between the hexadecimal, the binary and the ASCII code:

Binary	00110 0001	0110 0011	0111 1000
Hexadecimal	61	63	74
ASCII	'a'	'c'	't'

Binary files may contain any data whatsoever such as plain text, images, sound, compressed versions of other files (of either type), etc. (Wikipedia)
They are also used to store data output by a program, and intended to be read by that or another program but not by humans. (Dictionary.com)

Advantages
The advantages of binary files are:

- quicker to read (compared to plain text files)
- better for larger datasets

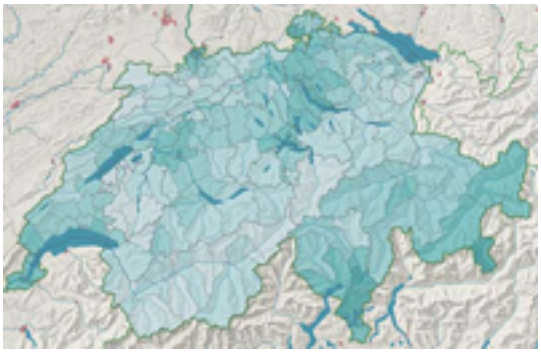
Disadvantages
The disadvantages of binary files are:

- not readable by humans
- fixed datastructure

Furthermore it is usually necessary to write special purpose programs to manipulate such files since most general purpose utilities operate on text files. (Dictionary.com)

Usage Scenarios

As we mentioned above, binary files are used to store raster data such as images or terrain models.



Picture saved as binary file (Atlas of Switzerland 2.0 2004)

[4] A byte comprises 8 bits. Since one bit can adopt two states it is possible to describe 256 (28) signs with one byte.

[5] A counting system that uses 16 digits, notated as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

1.1.3. XML (eXtensible Markup Language)

The markup language XML will be introduced in the Units **XML Introduction** and **XML Syntax** of this lesson. Therefore, we will not go into details of the syntax and the structure of XML now.

Example of an XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<station>
    <station_id>Jungfraujoch</station_id>
    <temperature>-5</temperature>
    <air_pressure>950</air_pressure>
    <wind_direction>270</wind_direction>
    <storm_force>43.7</storm_force>
</station>
```

Example of an XML file

Advantages

As you can see in the example, XML is text based. Humans are therefore able to read the file content.
There are some more advantages of XML:

- XML is platform and software independent
- The elements and attributes are easy to access
- XML is structured and extensible

XML can describe the structure of heterogeneous data.

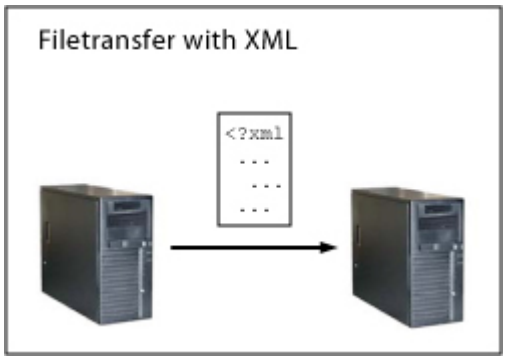
Disadvantages

The disadvantages of XML are:

- Bigger Filesize
- XML is not suitable for huge amounts of data, since currently no efficient management tools for larger XML files exist
- XML is not suitable for the storage of raster images

Usage Scenario

XML is currently very widespread. Every major software vendor and many open source projects support XML in one or more ways. Since XML is software and platform independent it is suitable for the data exchange between different systems.

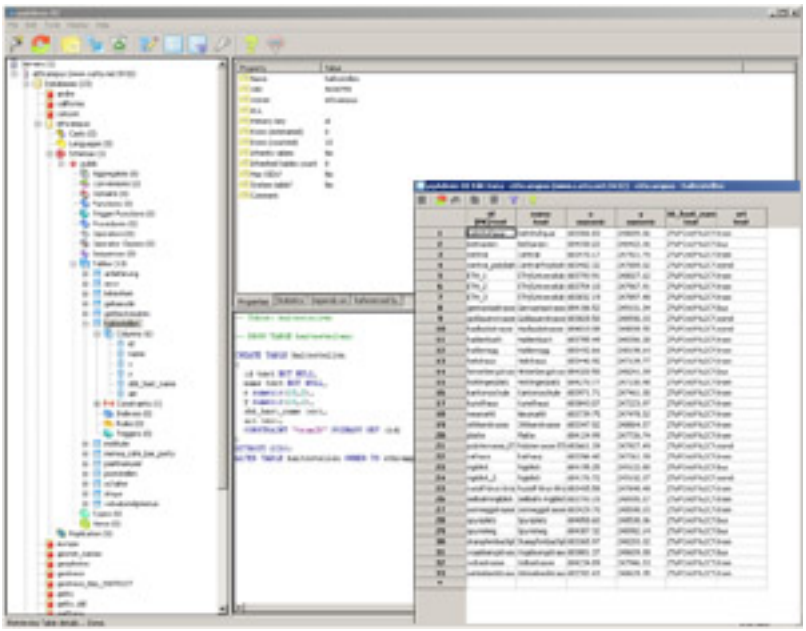


Filetransfer with XML

1.1.4. Databases

A database is an organized collection of information. The information is collected within tables which consist of rows and lines. There exist various database products:

- Open Source Databases:
  - PostgreSQL
  - MySQL
  - Firebird
- Commercial Databases:
  - Oracle
  - IBM DB2
  - MS SQL Server
  - Informix
  - MS Access



Example of a Database Software (PgAdmin III)

The open source databases are getting serious competitors to the commercial databases. In many cases it is easier and cheaper to use Open Source products compared to the more complex and expensive commercial ones.
Typically, for a given database, there is a structural description (also known as schema) of the type of information held in that database. The schema describes the objects that are represented in the database, and the relationships among them.
Databases are optimized for quick access and manipulation. Typically, the data of a database is created, modified and retrieved with SQL (Structured Query Language).
If you are interested in the functions and syntax of SQL, have a look at the W3C Tutorial **"Learn SQL"**

Advantages
The advantages of databases are:

- High performance,
- Quick and easy accessed,
- Standardized access through SQL,
- Can manage huge amounts of data,
- Multi-User support, Access control,
- The data within a database is structured,
- Can store different data types.

Disadvantages
The disadvantages are that the database product needs to be installed and there is also a need in administration skills for the software. The installation and the creation of the data tables is time consuming because there are several steps to create the final tables and their structure (creation of table, definition of attributes within the tables, filling the tables with data, etc.).

Databases are not suitable for the storage of a huge amount of binary files.

Usage Scenario
Databases are used for bigger until huge datasets which require an efficient data management.

1.1.5. Data Structure

We have several possibilities how to structure data. We can arrange the data in different orders:

- **by theme** (e.g. flora, fauna, meteorology, etc.)

- **by data type** (text, image, multimedia, etc.)

- **by time** (1960, 1970, 1980, etc.)

- etc.

In the last chapter we presented you four possibilities to store data. But we did not say much about the structure of these storage types. The following list summarises the structure and organisation of the four storage types. By clicking on the thumbnails you get an enlarged example of each type.

Exercise
You have a dataset with private data of various persons. For each person the following attributes are given: first name, last name, street, number, postal code, city, birthday, sex, telephone number, e-mail address, country, marital status, number of children, and nationality.
We want you to structure and group this data. Create a data schema (sketchily) which shows the structure of the data. Give reasons why you have chosen your structure. Choose a storage type that is suitable to the created data structure.
Hand in your data schema with your explanations to your tutor.

1.1.6. Unit-Summary

There exist four main file types which are used to store data:

- Plain Text Files: Simple data structure
- Binary Files: Fixed data structure
- XML: Simple - complex data structure
- Databases: Simple - complex data structure

Each file type has its advantages and disadvantages. Depending on the planned project certain file types are more qualified than others:

- Plain text files are mainly used for small datasets or for the import or export to or from spreadsheets programs or for the import into GIS or Mapping Software.
- Binary files are mostly used to store raster data such as images or terrain models.
- XML is mainly used for both small and large datasets but not for a huge amount of data. XML is suitable for the data exchange between two different systems such as databases, applications, etc.

- Databases are used for bigger until huge datasets which require an efficient data management.

Data can be structured in many different ways:

- by theme
- by data type
- by time
- etc.

Depending on the given data and purpose of it, an optimal data structure has to be chosen.

1.2. XML Introduction

Learning Objectives

You will be able...

- ...to explain the basic concepts of XML in three points.
- ...to list at least three properties of XML.

Introduction

XML is a markup language that features many advantages. Today, XML is very widespread because in 1998 it became a *W3C (World Wide Web Consortium)* [6] recommendation. Various technologies such as *XHTML* [7], *SVG* [8], etc. are based on XML. The purpose of XML is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. In addition, XML allows to structure your data in an optimal way. That is why we want to introduce XML in this lesson.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<introduction>
     <filetype>This file is
               written in XML</filetype>
     <easy>You will see that XML
           is very easy to learn</easy>
</introduction>
```

XML is easy to learn

In this unit we give you an overview of the properties and the features of XML. In further units you will learn more about the XML's syntax.
For this and the further unit you should have a basic understanding of HTML such as what is a tag and what it is used for, etc. If you want to study this subject first, find a tutorial on the **W3C Schools homepage**.

1.2.1. Definition and Example

XML stands for eXtensible Markup Language and is a simple, very flexible text format derived from *SGML* [9]. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. (WWW-Consortium)

[6] The World Wide Web Consortium (W3C) is the international standards body and develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential as a forum for information, commerce, communication, and collective understanding.

[7] Extensible Hypertext Markup Language, or XHTML, is a markup language that has the same expressive possibilities as HTML, but a stricter syntax.

[8] Scalable Vector Graphics (SVG) is an XML markup language for describing two-dimensional vector graphics, both static and animated.

[9] Abbreviation of Standard Generalized Markup Language, a system for organizing and tagging elements of a document. SGML was developed and standardized by the International Organization for Standards (ISO). SGML itself does not specify any particular formatting; rather, it specifies the rules for tagging elements. These tags can then be interpreted to format elements in different ways.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

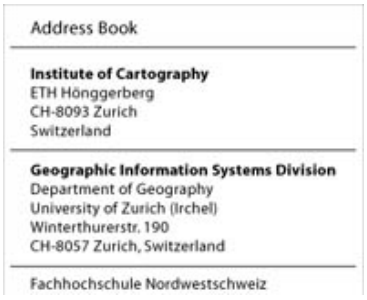
Example of an XML file

1.2.2. XML in 3 Points

The following summary in three points is taken out of a **W3C Homepage** and attempts to capture enough of the basic concepts to enable you to see the advantages of XML.

- **XML is for structuring data**
  Structured data includes things like spreadsheets, address books, technical drawings, etc.

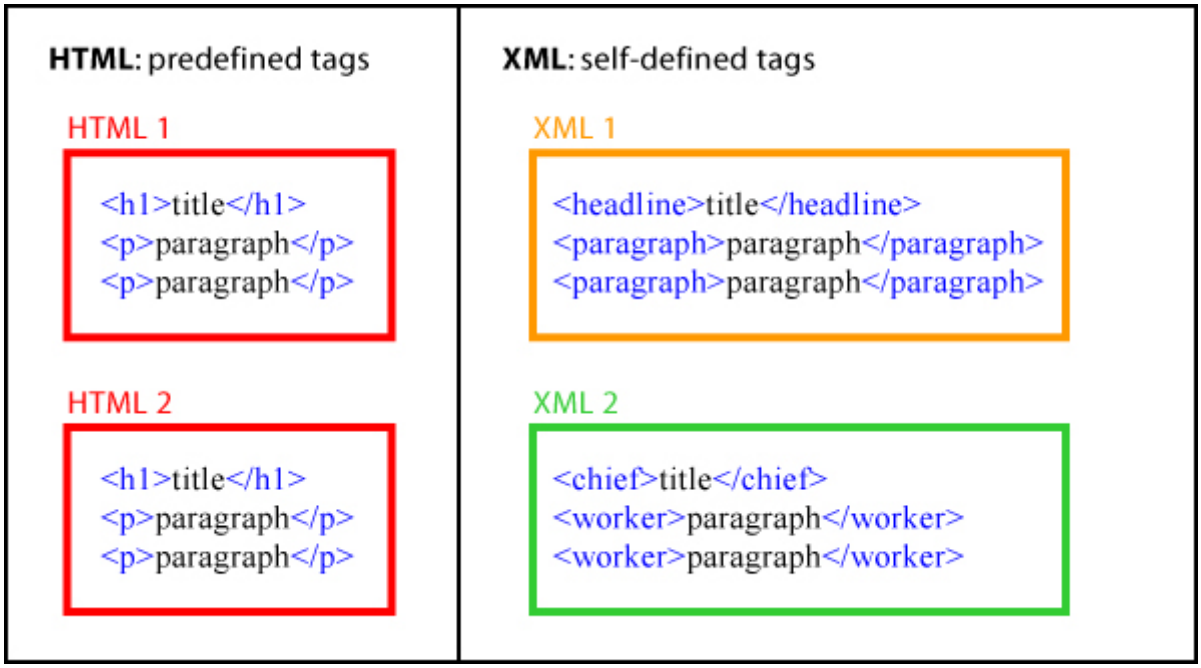
Spreadsheet



Address Book

Technical Drawing, © GIS-Fachstelle Kanton Basel-Landschaft (Kanton Basellandschaft (CH))

- XML is a set of rules (you may also think of them as guidelines or conventions) for designing text formats that let you structure your data. XML is not a programming language, and you do not have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous. XML is extensible, platform-independent, and it supports internationalization and localization.
- **XML looks a bit like HTML**
  Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"). While HTML specifies what each tag and attribute means, and often how the text between them will look in a browser, XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. In other words, if you see "<p>" in an XML file, do not assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person ... (and who says it has to be a word with a "p"?).



HTML tags versus XML tags

- **XML is text, but is not meant to be read**
  Like HTML, XML files are text files that people should not have to read, but may when the need arises. One advantage of a text format is that it allows people, if necessary, to look at the data without the program that produced it; that means, you can read a text format with your favourite text editor.
  Compared to HTML, the rules for XML files allow fewer variations. A forgotten tag, or an attribute without quotes makes an XML file unusable, while in HTML such practice is often explicitly allowed.

1.2.3. Properties and Features

The fact that XML is text-based is very important. You do not need a specific software installed to make small changes to your file. Although a full software suite might be more efficient to generate files or make larger changes, in many cases a text editor is good enough to make small changes.
The following statements which explain best the properties and features of XML are taken of the **W3Schools XML Tutorial**:

The main difference between XML and HTML
XML was designed to carry data and is not a replacement for HTML.
XML and HTML were designed with different goals:
XML was designed to describe data and to focus on what data is.
HTML was designed to display data and to focus on how data looks.
HTML is about displaying information, while XML is about describing information.

XML does not DO anything
XML was not designed to DO anything. XML was created to structure, store and to send information.
The following example is a note to Tove from Jani stored as XML:

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

A Note "written" in XML

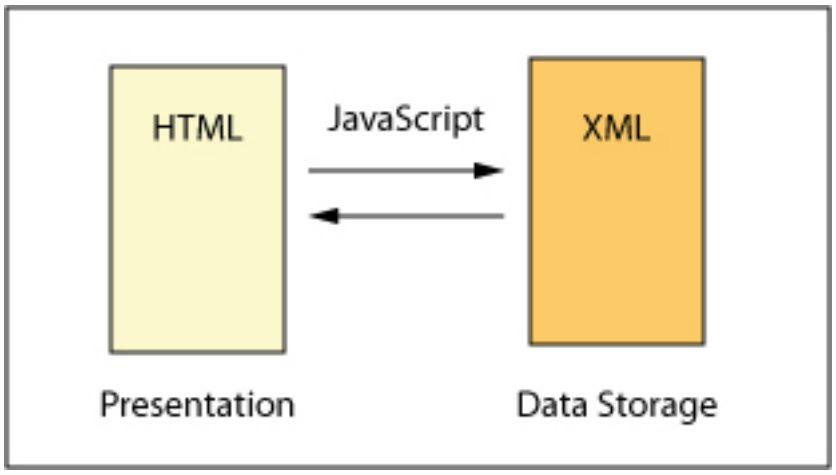
The note has a header and a message body. It also has sender and receiver information. But still, this XML document does not DO anything. It is just pure information wrapped in XML tags. Someone must write a piece of software to send, receive or display it.

XML is free and extensible

XML tags are not predefined. You must “invent” your own tags.
The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard (like <p>, <h1>, etc.).
XML allows the author to define his own tags and his own document structure.
The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

XML can separate data from Presentation

When HTML is used to display data, the data is stored inside your HTML. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML. To establish the connection between the presentation file (HTML) and the data storage file (XML) a client-side scripting language (such as JavaScript) can be used. Since the HTML then becames a dynamic characteristic, the HTML is called DHTML (Dynamic HTML).

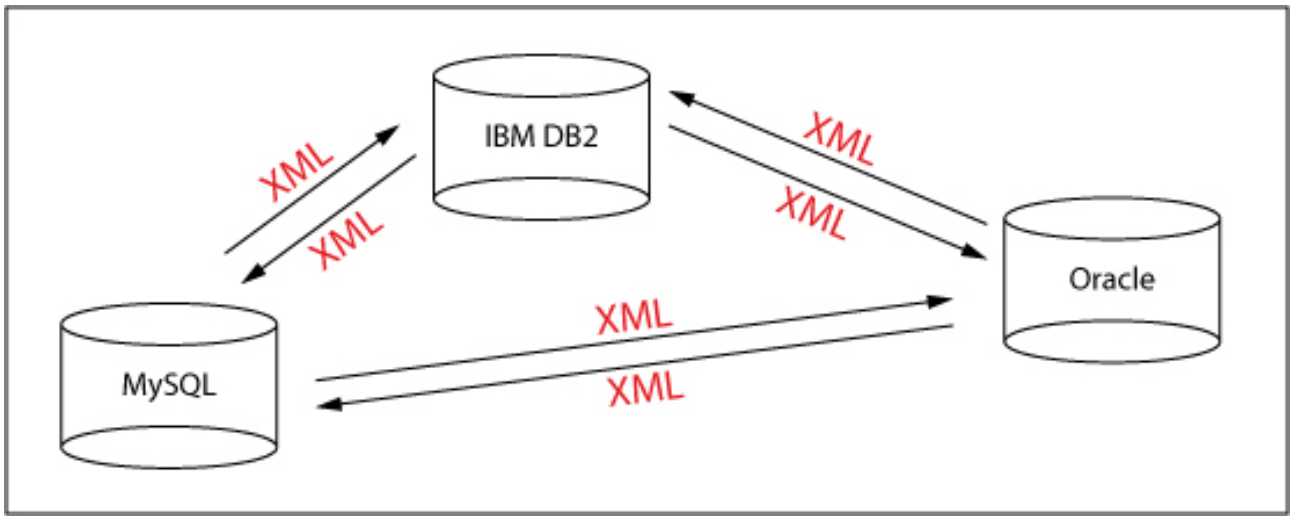


XML separates data from presentation

XML is used to exchange Data

In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.

Converting data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.



An example of how to exchange data with XML

A problem within this subject is that in XML tags can be defined individually, e.g. the tags <*title*> and <*Title*> are not the same even if the meaning of these tags might be the same. If an application would not accept the tag <*title*>, a converter would have to be created that converts all the tags <*title*> into the tag <*Title*>.

1.2.4. Unit-Summary

XML stands for eXtensible Markup Language and is a simple, very flexible text format. The basic concepts of XML can be summarised in three points:

- **XML is for structuring data**: XML is a set of rules for designing text formats that let you structure your data. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous.
- **XML looks a bit like HTML**: Like HTML, XML makes use of tags (words bracketed by '<' and '>'). Unlike HTML that specifies what each tag means, XML uses tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it.
- **XML is text, but is not meant to be read**: XML files are text files that people should not have to read, but may when the need arises.

XML has several properties and features:

- **XML does not DO anything**: XML was not designed to DO anything. XML was created to structure, store and to send information.
- **XML is free and extensible**: XML tags are not predefined. You must "invent" your own tags. XML allows the author to define his own tags and his own document structure.
- **XML can separate Data from presentation**: HTML is used to display data and XML is used to store data.
- **XML is used to Exchange Data**: XML facilitates the sharing of data across different systems, particularly systems connected via the Internet.

1.3. XML Syntax

Learning Objectives

You will be able...

- ...to name at least two rules that have to be followed when using tags in an XML document.
- ...to list the four relationships that XML tags can have.
- ...to give examples of forbidden, not recommended and allowed tag names.

Introduction

We introduced the basic concepts of XML in the last chapter. We already presented a simple XML example without any explanations about the syntax of the elements. We now want to give you an overview of the XML's elements and its syntax.
You will see that the syntax rules of XML are very simple and very strict.
As you have already learned, in XML you have to "invent" your own tags. In this unit we will present you which tag names are allowed, not recommended or even forbidden.

```
<root>
    <child>
        <subchild>....</subchild>
    </child>
</root>
```

XML's elements

This unit is based on the **XML Tutorial** of the W3Schoools. If you are interested in more details about XML have a look at that tutorial.

1.3.1. Example XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<address>
    <first_name>Peter</first_name>
    <last_name>Miller</last_name>
    <street>Hauptstrasse</street>
    <number>20</number>
    <city>Zurich</city>
</address>
```

Example XML Document

The first line in the document – the XML declaration "<?xml version="1.0" encoding="ISO-8859-1"?>" – defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) *character set* [10].
The next line describes the root element of the document:

[10] An encoding scheme in which each character is represented by a different binary value. For example, ISO8859-1 is an extended Latin character set that supports more than 40 Western European languages.

```
<address>
```

Root Element

The next four lines describe child elements (first_name, last_name, street, number, and city) of the root :

```
<first_name>Peter</first_name>
<last_name>Miller</last_name>
<street>Hauptstrasse</street>
<number>20</number>
<city>Zurich</city>
```

Child Elements

And finally the last line defines the end of the root element:

```
</address>
```

End of the Root Element

Can you detect from this example that the XML document contains an address? Don't you agree that XML is pretty self-descriptive?

1.3.2. XML Tags

All XML elements must have a closing tag

With XML it is illegal to omit the closing tag. As you perhaps know in HTML some elements do not have a closing tag such as e.g. "<p>". The following code is legal in HTML:

```
<p>This is the first paragraph
<p>This is the second paragraph
```

Legal HTML Code

In XML however, all elements have a closing tag, like this:

```
<p>This is the first paragraph</p>
<p>This is the second paragraph</p>
```

Legal XML Code

You might have noticed from the previous example that the XML declaration (<?xml version="1.0" encoding="ISO-8859-1"?>) did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it therefore does not need a closing tag.

All XML elements must be properly nested

All XML elements must be properly nested within each other: you are not allowed to close an XML tag until other tags that are enclosed by that tag (so called childs), are closed as well:

```
<text><bold>This text is NOT properly nested</text></bold>
<text><bold>This text is properly nested</bold></text>
```

Tag nesting

XML tags are case sensitive

Opening and closing tags must be written with the same case. The tag <message> is different from the tag <Message>:

```
<Message>Incorrect</message>
<message>Correct</message>
```

Tags are case sensitive

All XML documents must have a root element

All XML documents must contain a single tag pair to define a root element. The name of the root element must appear **once** in an XML document. All other elements must be within this root element. All elements can have sub elements (child elements). Sub elements must be correctly nested within their parent element:

```
<root>
    <child>
        <subchild>....</subchild>
    </child>
</root>
```

Root element and its children

By expanding our address example with the root element <person> we get the same structure as above (root, child, subchild).

```
<person>
    <address>
        <first_name>Peter</first_name>
        <last_name>Miller</last_name>
        <street>Hauptstrasse</street>
        <number>20</number>
        <city>Zurich</city>
    </address>
</person>
```

Expanded address example

Comments in XML

The syntax for writing comments in XML is:

```
<!--This is a comment-->
```

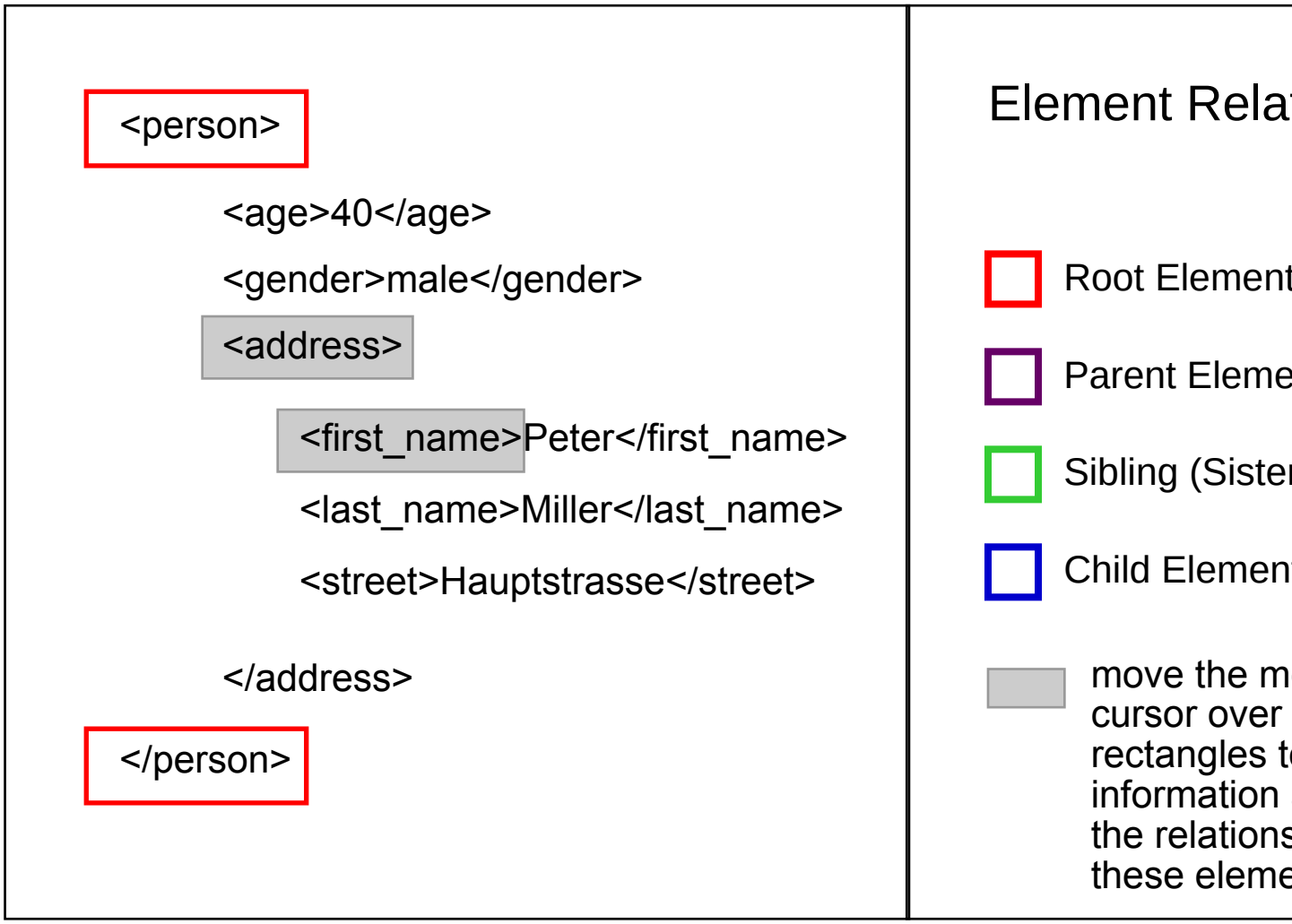
Comments in XML

1.3.3. XML Elements

XML elements are extensible and they have relationships.

XML elements have relationships

To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described. We still look at the Person Example:



As you can see XML elements are always related as parents and children.

Exercise
Open the following XML-document in a browser and have a look at it. Define the root, parent, child and sister elements of the XML-file. Hand in your results to the tutor.
Link to the XML-File: **cartouche.xml**

XML elements are extensible
Look at the following XML example:

```
<person>
    <address>
        <first_name>Peter</first_name>
        <last_name>Miller</last_name>
        <street>Hauptstrasse</street>
        <number>20</number>
        <city>Zurich</city>
    </address>
</person>
```

An XML Example

Let's imagine that we created an application that extracted the <first_name>, <last_name>, <street>, <number>, and <city> elements from the XML document to produce this output:

Application Output

Imagine that the author of the XML document added some extra information to it:

```
<person>
    <age>40</age>
    <gender>male</gender>
    <address>
        <first_name>Peter</first_name>
        <last_name>Miller</last_name>
        <street>Hauptstrasse</street>
        <number>20</number>
        <city>Zurich</city>
    </address>
</person>
```

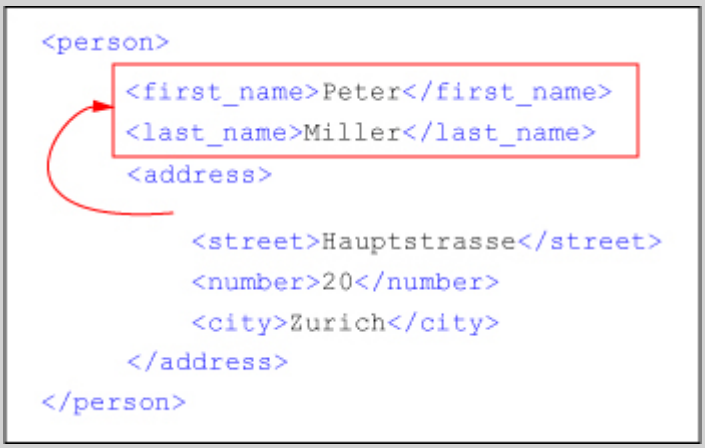
Expanded XML Example

Will the application break or crash? What do you think? The answer is given in the popup window below.

No. The application should still be able to find the <first_name>, <last_name>, <street>, <number>, and <city> elements in the XML document and produce the same output. Of course it depends on the structure and commands of the application. But with the right commands, the application is still able to find the right tags.
When would the application crash?
The application would crash if we would change the tag names of the extracted elements or the nesting structure of the elements as it is shown in the following pictures:

```
<person_data>
    <address>
        <firstname>Peter</firstname>
        <lastname>Miller</lastname>
        <Street>Hauptstrasse</Street>
        <Number>20</Number>
        <City>Zurich</City>
    </address>
</person_data>
```

Changed tag names



Changed nesting structure

1.3.4. XML Attributes

XML elements can have attributes in the start tag. They are used to provide additional information about elements. Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">red_deer.gif</file>
```

XML Attribute

Attributes values must always be enclosed in quotes, but either single or double quotes can be used:

```
<file type="gif">red_deer.gif</file>
```

XML Attribute with double quotes

```
<file type='gif'>red_deer.gif</file>
```

XML Attribute with single quotes

Use of elements versus attributes

Data can be stored in child elements or in attributes.

```
<person sex="male">
    <first_name>Peter</first_name>
    <last_name>Miller</last_name>
</person>
```

Information stored as attribute

```
<person>
    <sex>male</sex>
    <first_name>Peter</first_name>
    <last_name>Miller</last_name>
</person>
```

Information stored as element

In the first example the sex is an attribute. In the last, sex is a child element. Both examples provide the same information. There are no strict rules whether to use attributes or child elements. It is recommended though to use elements to describe data itself and attributes only for metadata (information that describes the data).

Exercise
Imagine you want to store the data of a map in an XML document. Which elements have to be included in this document and how should the structure of these elements look like?
Possible elements could be: map scale, width and height of the map, projection, legend, title, subtitle, relief and other map layers, etc.
Create an XML document that includes all elements that are needed for a map. Which elements are included as attributes and which ones as elements? Hand in your XML document to your tutor.

1.3.5. Element Naming and Wellformedness

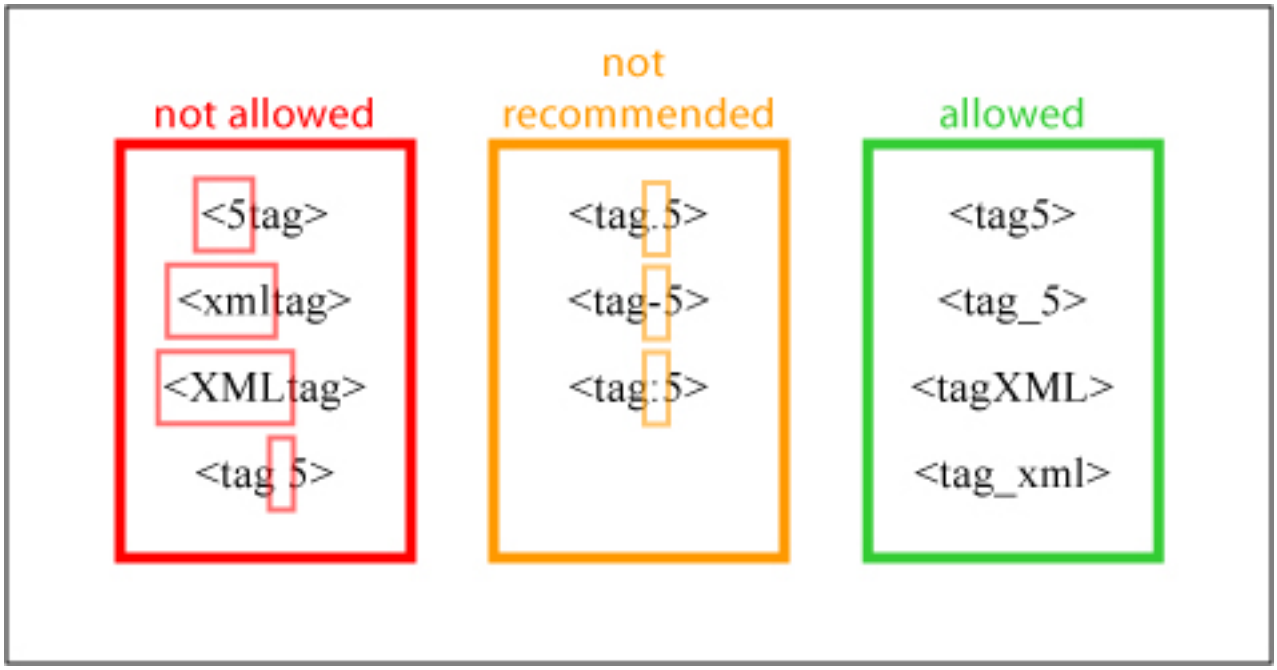
XML elements have simple naming rules as you will see in this chapter.
XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names **must not start** with a number or punctuation characters. Names **must not start** with the letters xml (or XML, or Xml, etc.)
- Names **cannot contain** spaces.

Take care when you "invent" element names and follow these simple rules:

- Any name can be used, no words are reserved, but the idea is to make names descriptive.
- **Avoid** " **-** " and " **.** " in names such as <first-name> or <first.name>.
- Element names can be as long as you like, but do not exaggerate. Ideally names are as short and simple as possible: <book_title> instead of <the_title_of_the_book>.
- Non-English letters like é ò á are perfectly legal in XML element names, but watch out for problems if your software vendor does not support them.

- The " **:** " **should not** be used in element names because it is reserved for *namespaces* [11].



Legal and illegal element names

"Well Formed" XML Documents

When having created an XML file you have to check if it is well formed. A "well formed" XML document is a document that conforms to the XML syntax rules above.

A check for wellformedness contains the following checks:

- if prolog <?xml version="1.0"?> is existing and valid,
- whether we have exactly one root element (e.g. <person>),
- if all elements are correctly opened and closed (<person> </person>),
- if the elements are correctly nested (<person><name> </name></person>).

Presentation of XML code

Spaces that are not part of the XML content (outside of tags) are ignored by an XML interpreter. Therefore, you are free in the arrangement of your XML code. It does not matter for an XML interpreter wheter the code is written all in one line or one line below the other without indentions, but from the point of view of a human XML-file reader, we stronlgy recommend to arrange your XML code in a legible way by using indentions.

The following figure shows two examples of hardly readable code and in contrast code that is easy to read for a human:

[11] XML Namespaces provide a method to avoid element name conflicts.

Code that is difficult to read by humans:

```
<person><address><name>Miller</name></address></person>

<person>
<address>
<name>Miller</name>
</address>
</person>
```

Code that is easy to read by humans:

```
<person>
    <address>
        <name>Miller</name>
    </address>
</person>
```

Presentation of XML code

XML editors format XML code automatically in a legible way.

Exercise
Check your XML document that you created in the last chapter for its wellformedness by opening in a web browser. The browser checks the document for its wellformedness and alerts the mistakes if your document is not wellformed. If you have questions ask your tutor.

1.3.6. Unit-Summary

The syntax rules of XML are very simple and very strict.
As we already told you, XML uses tags. There exist a few rules that have to be applied when using tags in an XML document:

- All XML elements must have a closing tag
- All XML elements must be properly nested
- XML tags are case sensitive
- All XML documents must have a root element

An XML document must have a root element and within this root element there can be as many nested elements as one pleases.
XML elements have relationships that are of the following four types:

- Root Element
- Parent Element

- Sibling (Sister Element)
- Child Element

As we told you in the former unit, XML allows the author to define his own tags and his own document structure. When naming the XML elements the following rules have to be followed:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation characters. Names must not start with the letters xml (or XML, or Xml, etc.)
- Names cannot contain spaces.

1.4. XML DTD and XML Schema

Learning Objectives

You will be able...

- ...to distinguish between internal and external DOCTYPE decalarations.
- ...to identify the reference to an XML Schema in an XML document when looking at a given XML document.
- ...to list the main differences between XML DTD and XML Schema.

Introduction

Until now, you learned how to create an XML document and how to "invent" XML tags. But we still did not tell you anything about how to define the structure and the elements that can or have to be used in a specific XML document. We make an example:
Imagine that you want to store the addresses of all your friends in an XML document and every address must have the same child elements such as first name, last name, street, number and city as it is shown in the following example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<address>
    <first_name>Peter</first_name>
    <last_name>Miller</last_name>
    <street>Hauptstrasse</street>
    <number>20</number>
    <city>Zurich</city>
</address>
```

Example XML Document

Don't you agree that somewhere, we have to define that the element <address> must have the child elements <first_name>, <last_name>, <street>, <number> and <city>, do you?
The specification of which elements are allowed in your document and what structure do they have is part of the DOCTYPE declaration (DTD) and the XML Schema.
If there exist a DTD or an XML Schema for the example above, you are not anymore able to store an address in the wrong way as it is shown in the next picture:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<first_name>Peter</first_name>
<last_name>Miller</last_name>
<address>
    <street>Hauptstrasse</street>
    <number>20</number>
    <city>Zurich</city>
</address>
```

Wrong tag nesting

The example shows you that DTD and XML Schema play an important role in the subject of XML. Therefore, we introduce the properties and the usage of DTD and XML Schema in this unit.
This unit is based on the **DTD Tutorial** and the **XML Schema Tutorial** of the W3C. If you are interested in more details about DTD and XML Schema have a look at those tutorials.
In addition to just introduce XML DTD and XML Schema, we will present you the main differences between these two schema types.

1.4.1. XML DTD

A Document Type Definition defines the legal building blocks of an XML document with a list of legal elements.
A DTD includes the following XML-structure and -syntax elements:

- list of valid elements,
- list of valid attributes,
- list of entities,
- definition on how elements can be nested,
- definition whether an element or attribute is required, recommended or optional,
- definition on how often elements may be used (0, 1 or more),
- definition of default values.

A DTD can be declared inline in your XML document, or as an external reference.

Internal DOCTYPE declaration
If the DTD is included in your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<?xml version="1.0"?>
<!DOCTYPE person [
    <!ELEMENT person  (age,gender,name)>
    <!ELEMENT age  (#CDATA)>
    <!ELEMENT gender  (#PCDATA)>
    <!ELEMENT name (#PCDATA)>
]>

<person>
  <age>40</age>
  <gender>male</gender>
  <name>Peter Miller</name>
</person>
```

Internal DOCTYPE declaration

The DTD above is interpreted like this:

- !DOCTYPE person (in line 2) defines that this is a document of the type person.
- !ELEMENT person (in line 3) defines the person element is having three elements: "age, gender, name".
- !ELEMENT age (in line 4) defines the age element to be of type "*#CDATA* [12]".
- !ELEMENT gender (in line 5) defines the gender element to be of type "*#PCDATA* [13]".
- And so on...

External DOCTYPE declaration

If the DTD is external to your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element SYSTEM "filename">
```

External DTD Declaration

The same XML document as above, but with an external DTD looks like this:

[12] CDATA means character data. CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will NOT be expanded.

[13] PCDATA means parsed character data. PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded.

```
<?xml version="1.0"?>
<!DOCTYPE person SYSTEM "person.dtd">

<person>
  <age>40</age>
  <gender>male</gender>
  <name>Peter Miller</name>
</person>
```

XML file with external DTD

And a copy of the file "person.dtd" containing the DTD looks like this:

```
<!ELEMENT person  (age,gender,name)>
<!ELEMENT age  (#CDATA)>
<!ELEMENT gender  (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

Content of the external DTD file "person.dtd"

Perhaps you are asking why do we have to use a DTD? Here are the answers:

- With DTD, each of your XML files can carry a description of its own format with it.
- With a DTD, independent groups of people can agree to use a common DTD for interchanging data.
- Your application can use a standard DTD to verify that the data you receive from the outside world is valid.
- You can also use a DTD to verify your own data.

If you are interested in more information about DTD (e.g. how to define attributes in a DTD document) consult the W3C Tutorial **"Learn DTD"**.
In this chapter we dealt with the "DTD Introduction"-part of this tutorial. Therefore you might continue with the chapter "DTD Building Blocks".

1.4.2. XML Schema

In this chapter we mention XML namespaces. If you do not know namespaces and their functions and you want to get familiar with this topic, have a look at the W3C Tutorial **"XML Namespaces"**. There, you will find the needed information.
The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.
An XML Schema defines:

- elements that can appear in a document,
- attributes that can appear in a document,
- which elements are child elements,

- the order of child elements,
- the number of child elements,
- whether an element is empty or can include text,
- data types for elements and attributes,
- default and fixed values for elements and attributes.

What do XML Schemas look like?
We introduce you the XML Schema corresponding to the following XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<person>
      <age>40</age>
      <gender>male</gender>
      <name>Peter</name>
</person>
```

XML Document

As you learnt in a former chapter, the corresponding DTD file "person.dtd" looks like this:

```
<!ELEMENT person  (age,gender,name)>
<!ELEMENT age  (#CDATA)>
<!ELEMENT gender  (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```

Content of the DTD file "person.dtd"

The XML Schema file called "person.xsd" that defines the elements of the XML document above, looks like this (move mouse over the coloured rectangles to get information about the functions of the different elements):

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.e-cartouche.ch/XMLSchema"
targetNamespace="http://www.e-cartouche.ch"
xmlns="http://www.e-cartouche.ch"
elementFormDefault="qualified">

<xs:element name="person">
    <xs:complexType>
      <xs:sequence>
      <xs:element name="age" type="xs:integer"/>
      <xs:element name="gender" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

The **person** element is said to be of a **complex type** because it contains other elements. The other elements (age, gender, name) are said to be **simple types** because they do not contain other elements.

The reference to an XML Schema in an XML document looks as following (move mouse over the coloured rectangles to get information about the functions of the different elements):

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

<person  xmlns="http://www.e-cartouche.ch"
xmlns:xsi="http://ww.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.e-cartouche.ch/person.xsd" >

<person>
        <age>40</age>
        <gender>male</gender>
        <name>Peter Miller</name>
</person>
```

If you want to get familiar with the definition of simple XML Schema elements or attributes have a look at the W3C Tutorial **"Learn XML Schema"** chapters **"XSD Elements"** and **"XSD Attributes"**. Of course, you are free to watch the other chapters as well ;-).

1.4.3. "Valid" XML documents

"Valid" XML documents
An XML document has to be checked not only for its wellformedness but also for its validation. We already presented you the wellformedness of an XML document in the former chapter. Just to remind you: A "well formed" XML document is a document that conforms to the XML syntax.
A "valid" XML document however is a "well formed" XML document, which also conforms to the rules of a DTD or an XML Schema.

1.4.4. DTD versus XML Schema

Both DTD and XML Schema are used to structure data and validate the XML syntax.
In the future, XML Schema might be used as a replacement for DTD's because of the following reasons (according to the **W3Schools XML Schema Tutorial**) :
XML Schemas...

- ...are extensible to future additions,
- ...are richer and more useful than DTDs,
- ...are written in XML,
- ...support simple (integer, decimal, string, boolean, dateTime, etc.) and complex data types ,
- ...support namespaces.

Because XML Schemas are written in XML they possess the same advantages as XML documents (listed in chapter **Introduction to XML**).

XML Schema provides more precise control over the text content of elements and attributes and their occurance than DTDs:

- Element occurance constraints
  Maximum and minimum number of times an element can occur (minOccurs, maxOccurs)
- Simple (integer, decimal, string, boolean, dateTime, etc.) and complex data types

```
<xs:element name="age" type="xs:integer"/>
```

Simple Data Type in a XML Schema

```
<xs:element name="employee">
  <xs:complexType>
    <xs:choice>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string" maxOccurs="1"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Complex Data Type in a XML Schema

- Explicit restrictions on the sequence of child elements.
  There are three possibilities how to control the order of the child elements:
  - **<xs: all>:** specifies that the child elements can appear in any order and that each child element must occur once and only once.
  - **<xs: choice>:** specifies that either one child element or another can occur.
  - **<xs: sequence>:** specifies that the child elements must appear in a specific order.
- Specific rules about the contents of elements and attributes

If you are interested in a list with the possible content restrictions for the content of elements and attributes, have a look at this popup-part

Restrictions for Datatypes

Constraints	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)

maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Restrictions for Datatypes (W3Schools)

However, DTDs do not provide fine control over the format and data types of element and attribute values. Once an element or attribute has been declared to contain character data, no limits may be placed on the length, type, or format of that content. (Harold et al. 2002)

1.4.5. Unit-Summary

DTD
A Document Type Definition defines the legal building blocks of an XML document with a list of legal elements. A DOCTYPE declaration can be internal or external.

XML Schema
An XML Schema defines the legal building blocks of an XML document, just like a DTD. The possibilities of XML Schemas are richer than the ones of DTDs.

XML Schema versus DTD
In the future, XML Schema might be used as a replacement for DTD's because XML Schemas...:

- ...are extensible to future additions,
- ...are richer and more useful than DTDs,
- ...are written in XML,
- ...support simple (integer, decimal, string, boolean, dateTime, etc.) and complex data types,

- ...support namespaces.

Because XML Schemas are written in XML, they possess the same advantages as XML documents.

1.5. Self Assessment

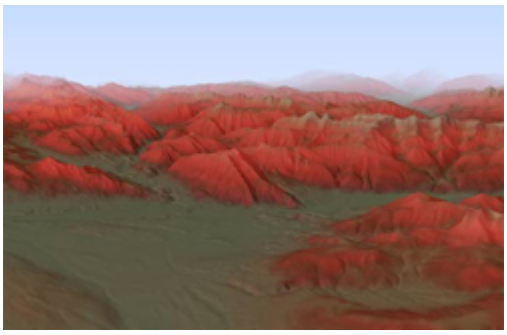
Data Storage

In the introduction of this lesson we showed you a few data examples and told you that after having studied this lesson, you should be able to define the best storage method for the given data examples:

Picture of Red Deer (Atlas of Switzerland 2.0 2004)

Red deer, often called king of the forest, reaches a head-body length of approx. 2 m, a shoulder height of up to 120 cm in females and 150 cm in males, as well as a weight of 90 to 120 kg, respectively. It is the largest mammal in Switzerland. Only the male (stag or bull) carries antlers which are shed and regrown each year...

Describing Text (Atlas of Switzerland 2.0 2004)



Picture of occurence of Red Deer (Atlas of Switzerland 2.0 2004)

X-Coord	Y-Coord	Z-Coord	#Red Deer
600050	200040	800	3
600060	200050	768	2
600070	200060	900	4
...	...	...	...

Table with Red Deer Occurence

We now want you to define the storage methods for the examples above (plain text file, ASCII, XML or database). Give reasons for your decision. Discuss your results on the discussion board "Data Storage" of this lesson and comment on at least three entries of your colleagues.

Hint: Think carefully about what type of data (text, image, etc.) the presented example is of.

XML Quiz

On the **W3Schools homepage** exists an XML Quiz where you can test your XML skills. The test contains 20 questions and there is no time limit. You will get 1 point for each correct answer. At the end of the Quiz, your total score will be displayed.

This Quiz is designed for the W3Schools XML tutorial which contains more information about XML than we presented here in this lesson. Therefore, you do not have to know the right answers for the following questions: 10, 18, 19 and 20. But you should be able to find the right answers for all the other questions.

Good Luck! **Start the XML Quiz**.

1.6. Summary

Data Storage

There exist several data storage methods and we presented you four of them. Depending on the type and the amount of data that has to be stored, the best storage method has to be found:

- Small datasets: Plain Text Files or XML
- Huge datasets: database
- Images and terrain models: binary files

Not only the amount of data but also the structure of given data define the decision for a storage type:

- Text Files: simple structure
- Binary Files: fixed structure
- XML: simple - complex structure
- Database: simple - complex structure

XML

XML is an open standard to structure, store and send information. It allows the author to define their own tags and their own document structure. XML is not to display data but to describe data and to focus on what data is. If you compare XML to HTML we can say, that HTML is used to display data and XML is used to carry and describe data.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<root>
    <child>
        <subchild>....</subchild>
    </child>
</root>
```

XML

The main purpose of XML is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. An XML document can therefore be transmitted and interpreted between applications and organizations.
In the majority of cases you do not have to create an XML file by your own, but the XML file is produced automatically by a computer program e.g. by exporting a document as XML file. Nevertheless you should be able to interpret the content of an XML file, even if it has been created automatically. This is the reason why we introduced XML in this lesson. And be aware that there are cases in which you have to create XML files by your own. By now you should know the XML syntax.

DTD and XML Schema

DTD and XML Schema contain the rules that specify the structure for particular XML documents. The structure is defined with elements, attributes, and notations. DTD and XML Schema establish constraints for how each element, attribute, and notation may be used within the particular documents.

1.7. Glossary

ASCII:

ASCII (American Standard Code for Information Interchange) is a character encoding based on the English alphabet. ASCII represents text in computers, communications equipment, and other devices that work with text. (Wikipedia)

Byte:

A byte comprises 8 bits. Since one bit can adopt two states it is possible to describe 256 (2^8) signs with one byte.

CDATA:

CDATA means character data. CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will NOT be expanded. (W3Schools)

Character Set:

An encoding scheme in which each character is represented by a different binary value. For example, ISO8859-1 is an extended Latin character set that supports more than 40 Western European languages. (Naude 2006)

Hexadecimal:

A counting system that uses 16 digits, notated as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Hyperlink:

An element in an electronic document that links to another place in the same document or to an entirely different document. Typically, you click on the hyperlink to follow the link. (Panama-Hosting.com) It is most commonly used in the World Wide Web to link various documents (Web Pages, pdf-files, etc.).

ISO:

The **International Organization for Standardization** is the international organisation responsible for developing and maintaining technical standards. (InterSites)

Namespace:

XML Namespaces provide a method to avoid element name conflicts. (W3Schools)

PCDATA:

PCDATA means parsed character data. PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded. (W3Schools)

SGML:

Abbreviation of Standard Generalized Markup Language, a system for organizing and tagging elements of a document. SGML was developed and standardized by the International Organization for Standards (ISO). SGML itself does not specify any particular formatting; rather, it specifies the rules for tagging elements. These tags can then be interpreted to format elements in different ways. (Borland)

SVG:

"*Scalable Vector Graphics (SVG) is an XML markup language for describing two-dimensional vector graphics, both static and animated.*" (Wikipedia)

W3C:

The World Wide Web Consortium (W3C) is the international standards body and develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential as a forum for information, commerce, communication, and collective understanding. (ETV Cookbook)

XHTML:

Extensible Hypertext Markup Language, or XHTML, is a markup language that has the same expressive possibilities as HTML, but a stricter syntax. (Wikipedia)

1.8. Bibliography

- **Atlas of Switzerland 2.0** (2004). *Atlas of Switzerland*. [CD-ROM]. Version 2.0. Switzerland: Swiss Federal Office of Topography (swisstopo).
  Download: http://www.atlasderschweiz.ch/index_en.html
- **Borland**. *Glossary* [online]. Available from: http://info.borland.com/techpubs/books/vbj/vbj40/programmers-guide/glossary.html [Accessed 10.01.2006].
- **Dictionary.com**. *Binary File* [online]. Available from: http://dictionary.reference.com/search?q=binary%20file [Accessed 16.01.2006].
- **ETV Cookbook**. *Glossary* [online]. Available from: http://etvcookbook.org/glossary/ [Accessed 19.10.2005].
- **Harold, E.R., Means, S.**, 2002. *XML in a Nutshell*. 2nd Edition. O'Reilly.
- **InterSites**. Available from: http://www.intersites.co.uk/7988/ [Accessed 10.01.2006].
- **Kanton Basellandschaft (CH)**. *Geoportal des Kantons Basel-Landschaft* [online]. Available from: http://www.geo.bl.ch/ [Accessed 09.02.2007].
- **Naude, F.** (2006). *Glossary* [online]. Available from: http://www.orafaq.com/glossary/faqglosc.htm [Accessed 25.04.2006].
- **Panama-Hosting.com**. *Glossary of Common Internet Terms* [online]. Available from: http://www.panama-hosting.com/glossary.htm [Accessed 19.10.2005, now dead link].
- **PgAdmin III**. *pgAdmin PostgreSQL Tools* [online]. Available from: http://www.pgadmin.org [Accessed 09.02.2007].
  Download: http://www.pgadmin.org/download/
- **syrma software**. Available from: http://www.syrma.de [Accessed 17.02.2006].
  Download: http://www.syrma.de/products/sqltools/screenshots/dbbrowser.png
- **W3Schools**. *XML Namespaces* [online]. Available from: http://www.w3schools.com/xml/xml_namespaces.asp [Accessed 10.01.2006].
- **W3Schools**. *DTD - XML building blocks* [online]. Available from: http://www.w3schools.com/dtd/dtd_building.asp [Accessed 10.01.2006].
- **W3Schools**. *DTD - XML building blocks* [online]. Available from: http://www.w3schools.com/dtd/dtd_building.asp [Accessed 10.01.2006].
- **W3Schools**. *Introduction to XML Schema* [online]. Available from: http://www.w3schools.com/schema/schema_intro.asp [Accessed 24.01.2006].
- **Whatis.com**. *Binary File* [online]. Available from: http://whatis.techtarget.com/definition/0,,sid9_gci213734,00.html [Accessed 16.01.2006].
- **Wikipedia**. *XHTML* [online]. Available from: http://en.wikipedia.org/wiki/XHTML [Accessed 17.02.2006].
- **Wikipedia**. *Scalable Vector Graphics* [online]. Available from: http://en.wikipedia.org/wiki/SVG [Accessed 05.12.2005].
- **Wikipedia**. *ASCII* [online]. Available from: http://en.wikipedia.org/wiki/ASCII [Accessed 10.01.2006].
- **Wikipedia**. *Binary and Text Files* [online]. Available from: http://en.wikipedia.org/wiki/Binary_and_text_files [Accessed 16.01.2006].
- **WWW-Consortium**. *Extensible Markup Language (XML)* [online]. Available from: http://www.w3.org/XML/ [Accessed 14.01.2006].
- **WWW-Consortium**. *XML in 10 Points* [online]. Available from: http://www.w3.org/XML/1999/XML-in-10-points.html [Accessed 14.01.2006].