

**X****M****L**

# Data Management

dott. Chris Mair  
2009/2010 @unibz

## 2 – Schema validation; the DTD

last modified 2009-10-14

# Schemas for XML

- a schema for XML is a grammar that defines a specific parser
- the parser can read the XML document as input and produce a validation report as output (boolean: the document is either valid or not)
- e.g. `telegram.dtd` – we briefly saw it last time

# Benefits of schemas

- a schema can act as a public specification, e.g. XHTML, Open Document Format
- quality control / input validation at a software interface, e.g. manual checking of HTTP parameters vs. automatic checking with WSDL
- portability, i.e. schemas substitute custom parsers (think yacc/bison)

# Kinds of schema

- **DTD** (document type definition, ancient)
  - **XML Schema** (from W3C)
  - **RELAX NG** (pronounced *relaxing*)
  - **Schematron**
- ▶ we'll see DTD and XML Schema during this course

# Schemas compared

Feature	DTD	W3C Schema	RELAX NG	Schematron
XML syntax	No	Yes	Yes	Yes
Namespace compatible	No	Yes	Yes	Yes
Declares entities	Yes	No	No	No
Tests datatypes	No	Yes	Yes	Yes
Default attribute values	Yes	Yes	No <sup>[7]</sup>	No
Notations	Yes	Yes	No	No
Unordered content	No	Yes	Yes	Yes
Modular	Yes	Yes	Yes	No
Element-attribute interchangeability	No	Yes	Yes	Yes
Specifies how to associate with a document	Yes	Yes	No	No

<sup>[7]</sup> Added later in the RELAX NG DTD Compatibility specification.

source: "Learning XML"

# Enter the DTD

course.xml

```
<?xml version="1.0"?>  
<!DOCTYPE course SYSTEM "course.dtd">  
<course>  
    XML Data Management  
</course>
```

*document type definition*

*document type declaration*

course.dtd

```
<!ELEMENT course (#PCDATA)>
```

# DTD principles

- a DTD declares a set of allowed **elements**
- you cannot use any elements other than those in this set
- a DTD establishes a pattern for each of the allowed elements
- the pattern defines what can be contained in an element (what other elements or character data):
  - in what order
  - in what number and
  - whether it's required or optional

# DTD principles cont'

- a DTD declares a set of allowed **attributes** for each element
- for each attribute it defines:
  - name
  - datatype (in a very limited way, unfortunately)
  - default value
  - whether it's required or optional

# The telegram DTD

```
<!ELEMENT telegram (from,to,subject,graphic?,message)>
<!ATTLIST telegram pri CDATA #IMPLIED>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT graphic EMPTY>
<!ATTLIST graphic fileref CDATA #REQUIRED>
<!ELEMENT message (#PCDATA|emphasis|name|villain)*>
<!ELEMENT emphasis (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT villain (#PCDATA)>
```

# The telegram DTD

```
<!ELEMENT telegram (from,to,subject,graphic?,message)>
```



There is an element `telegram` that must contain

- one element `from`,
- one element `to`,
- one element `subject`,
- zero or one element `graphic` and
- one element `message`

in precisely that order.

Note: DTDs cannot define which element is the document element (i.e. the root element)!

# The telegram DTD

```
<!ELEMENT telegram (from,to,subject,graphic?,message)>
```



Quantifiers are well known from regular expressions:

- ? zero or one
- \* zero or more
- + one or more

# The telegram DTD

```
<!ATTLIST telegram pri CDATA #IMPLIED>
```

The element `telegram` has one attribute, named `pri`. The attribute is of type character data (CDATA), i.e. a String and it is optional (`#IMPLIED`). The alternative to `#IMPLIED` is `#REQUIRED`.

# The telegram DTD

```
<!ELEMENT from (#PCDATA)>
```

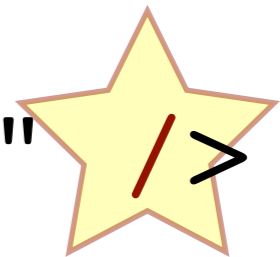
The element `from` contains just (parsed) character data (`#PCDATA`). *Parsed* meaning that entity references (think `&macro;`) are expanded.

# The telegram DTD

```
<!ELEMENT graphic EMPTY>
```

The element `graphic` is an empty element, e.g.:

```
<graphic fileref="figs/me.eps" />
```



# The telegram DTD

```
<!ELEMENT message (#PCDATA | emphasis | name | villain) *>
```

The element `message` is a typical **mixed-content type** element: it contains zero or more of *any* of the elements inside the brackets or character data. In this kind of rule, `#PCDATA` must come first and the brackets must be followed by `*`.

# Hands-on!

- we will meet after the coffee break in lab room **E331**
- try out the DTDs from the notes-2 sample files:  
`course.dtd`  
`telegram.dtd`  
`personal.dtd`