DTD

We saw at page 26 that the **Document Type Declaration** may contain markup which constrains the XML document it belongs to (elements, attributes etc.).

The content of a Document Type Declaration is made of a **Document Type Definition**, abbreviated DTD. So the former is the container of the latter.

It is possible to have all or part of the DTD in a separate file, usually with the extension "`.dtd`".

DTD/Attribute lists

We already saw **attribute lists** at page 41 when setting up internal linking.

In general, the `ATTLIST` can be used to specify any kind of attributes of an element, not just label and references.

Consider the attribute declarations for element `memo`:

```
<!ATTLIST memo
          ident      CDATA           #REQUIRED
          security   (high | low)    "high"
          keyword    NMTOKEN         #IMPLIED>
```

`CDATA` stands for **character data** and represents any string. A **named token** (`NMTOKEN`) is a string starting with a letter and which may contain letters, numbers and certain punctuation.

DTD/Element declarations

In order for a document to be validated, which requires more constraints than to be merely well-formed, all the elements used must be declared in the DTD.

The name of each element must be associated a **content model**, i.e., a description of what it is allowed to contain, in terms of textual data and sub-elements (mark-up).

This is achieved by means of the DTD `ELEMENT` declarations. There are five kinds of content models.

DTD/Element declarations (cont)

The first kind is the **empty element**:

```
<!ELEMENT padding EMPTY>
```

The second is elements with **no content restriction**:

```
<!ELEMENT open ALL>
```

The third is elements containing **only text**:

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

which means **parsed-character data**.

DTD/Element declarations (cont)

The fourth is elements containing **only elements**:

```
<!ELEMENT section (title, para+)>
<!ELEMENT chapter (title, section+)>
<!ELEMENT report (title, subtitle?, (section+ | chapter+))>
```

where `title`, `subtitle` and `para` are elements.

The last is elements containing **both text and elements**:

```
<!ELEMENT para (#PCDATA | emphasis | ref)+>
```

where `emphasis` and `ref` are element names.

DTD/Element declarations (cont)

The technique to define the content model is similar to **regular expressions**. Such an expression can be made up by combining the following:

- (`e1, e2, ..., en`) represents the elements represented by e_1, followed by the elements represented by e_2 etc. until e_n;
- e_1 | e_2 represents the elements represented by e_1 or e_2;
- (`e`) represents the elements represented by e;
- e? represents the elements represented by e or none;
- e+ represents the non-empty repetition of the elements represented by e;
- $e*$ represents the repetition of the elements represented by e.

DTD/Element declarations (cont)

Warning. When mixing text and elements, the only possible regular expression is either

```
(#PCDATA)
```

or

```
(#PCDATA | ...)*
```

DTD/Internal and external subsets

The part of a DTD which is included in the same file as the XML document it applies to is called the **internal subset**. See again the example page 44.

The part of a DTD which is in an independent file (`.dtd`) is called the **external subset**.

If there is no internal subset and everything is in the external subset we have a declaration like

```
<!DOCTYPE some_root_element SYSTEM "some.dtd">
```

DTD/Validating parsers

In order to validate an XML document, its DTD must completely describe the elements and attributes used.

This is not mandatory when well-formedness is required.

Therefore, the example page 44 is well-formed but not valid in the sense above, because the elements `map` and `country` are not declared.

For example:

```
<!ELEMENT map     (country+)>
<!ELEMENT country EMPTY>
```

would be sufficient to validate the document.