DTD (Document Type Definition)

In simple cases, **DTD** is used to describe the syntax of a data structure stored in XML. This DTD can be saved in a file with a `.dtd` extension, but it can also be part of the XML document itself.

Basic Properties of DTD

- Allows for simple **syntactic validation**.

- A descriptive language similar to **extended Backus-Naur form**.

- You can define structures, required sequences, type constraints, and cardinality.

If an XML document has a document type declaration, it must be indicated in the file with a special declaration starting with `<!DOCTYPE>`. The type declaration can be **internal** or **external** (referenced via **URI** or a **file**).

Internal declaration

```
<!DOCTYPE uzenet [
        ....
]>
```

External URI

```
<!DOCTYPE html PUBLIC "-//W3//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/......dtd">
```

External file

```
<!DOCTYPE uzenet SYSTEM "filename.dtd">
```

The DTD defines the structure and rules that an XML document must follow to be considered valid.

DTD Through Examples

Here is a DTD file named `message_syntax.dtd` and the corresponding XML on the right:

```
<!ELEMENT message ( text )>
<!ELEMENT text ( #PCDATA )>
```

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE message SYSTEM "message_syntax.dtd">

<message>
   <text>Hello XML</text>
</message>
```

In the XML structure, a **comma** indicates the required order of elements:

```
<!DOCTYPE studygroup [
        <!ELEMENT group (teacher, student)>
        <!ELEMENT teacher ( #PCDATA ) >
        <!ELEMENT student ( #PCDATA ) >
]>
```

```
<studygroup>
    <teacher>Kiss Janos</teacher>
    <student>Gipsz Jakab</student>
</studygroup>
```

The **pipe symbol** `|` expresses an **either-or** relationship. For example, the storage device can be either a **pendrive** or an **SSD**, but not both:

```
<!DOCTYPE datastorage [
    <!ELEMENT datastorage  (pendrive | SSD)>
    <!ELEMENT pendrive ( #PCDATA ) >
    <!ELEMENT SSD ( #PCDATA ) >
]>
```

```
<datastorage>
    <pendrive>64MB</pendrive>
</datastorage>
```

There are three ways to express **frequency**:

Symbol	Meaning
+	The element appears at least once.
*	The element can appear any number of times, including zero.
?	The element appears zero or one time.

Example: A music CD contains **tracks:**

```
<!DOCTYPE datastorage [
    <!ELEMENT CD (recording + )>
    <!ELEMENT recording ( #PCDATA ) >
]>
```

```
<CD>
    <felvetel>Song 1</felvetel>
    <felvetel>Song 2</felvetel>
</CD>
```

Example: An album has at least one title, followed by at least one **track title** and **duration**:

```dtd <!DOCTYPE adathordozo [

  <!ELEMENT album (cim+, (dalcim, idotartam)+)>
  <!ELEMENT cim ( #PCDATA ) >
  <!ELEMENT dalcim ( #PCDATA ) >
  <!ELEMENT idotartam ( #PCDATA ) >

]> ```

**Syntactically correct XML**:

```xml <album>

  <cim>Title 1</cim>
  <cim>Subtitle</cim>
  <dalcim>Track Title 1</dalcim>
  <idotartam>3.42</idotartam>
  <dalcim>Track Title 2</dalcim>
  <idotartam>2.32</idotartam>

</album> ```

#### Example: A library may contain **books** (zero or more):

```dtd <!DOCTYPE adathordozo [
```

```
  <!ELEMENT konyvtar (konyv*) >
  <!ELEMENT konyv (szerzo, cim) >
  <!ELEMENT szerzo ( #PCDATA ) >
  <!ELEMENT cim ( #PCDATA ) >
```

]> ```

Syntactically correct XML:

```xml <konyvtar>

```
  <konyv>
      <szerzo>Orwell, George</szerzo>
      <cim>1984</cim>
  </konyv>
  <konyv>
      <szerzo>Brown, Dan</szerzo>
      <cim>The Da Vinci Code</cim>
  </konyv>
```

</konyvtar> ```
```