

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 message [  
    ....  

```

External URI

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

External file

```
<!DOCTYPE message 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 datastore [  
  <!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>
  <recording>Song 1</recording>
  <recording>Song 2</recording>
</CD>
```

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

```
<!DOCTYPE datastorage [
  <!ELEMENT album (title+, (tracktitle, duration)+)>
  <!ELEMENT title ( #PCDATA ) >
  <!ELEMENT tracktitle ( #PCDATA ) >
  <!ELEMENT duration ( #PCDATA ) >
]>
```

Syntactically correct XML:

```
<album>
  <title>Title 1</title>
  <title>Subtitle</title>
  <tracktitle>Track Title 1</tracktitle>
  <duration>3.42</duration>
  <tracktitle>Track Title 2</tracktitle>
  <duration>2.32</duration>
</album>
```

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

```
<!DOCTYPE datastorage [  
  <!ELEMENT library (book*) >  
  <!ELEMENT book (author, title) >  
  <!ELEMENT author ( #PCDATA ) >  
  <!ELEMENT title ( #PCDATA ) >  
>
```

Syntactically correct XML:

```
<library>  
  <book >  
    <author>Orwell, George</author>  
    <title >1984</title >  
  </book>  
  <book>  
    <author>Brown, Dan</author>  
    <title >The Da Vinci Code</title >  
  </book >  
</library>
```

Defining Attributes in DTD

If a class has an attribute such as “number of students,” it can be specified as follows:

```
<!ELEMENT class (student *) >  
<!ATTLIST class number CDATA #REQUIRED>
```

Attributes can be: - **#IMPLIED**: not mandatory - **#REQUIRED**: mandatory - **#FIXED**: fixed value

Specifying a Default Attribute Value

Example:

```
<!ATTLIST paymentType type CDATA "bankTransfer">
```

```
</sxh
```

The XML could look like this:

```
<sxh>  
<paymentType/> or <paymentType type="bankTransfer">
```

Enumerated Values

Syntax:

```
<!ATTLIST element-name attribute-name (eval | eval | ..) default-value>
```

DTD Example:

```
<!ATTLIST payment type (check | cash) "cash">
```

XML Example:

```
<payment type="check"/> or <payment type="cash"/>
```

Complex DTD Example for a Hypothetical Mail Processing System

Task: Provide an XML example that satisfies the following DTD:

```
<!ELEMENT mails (email*, postcard*)>  
<!ELEMENT email (address, sender, message?, attachment?)>  
<!ELEMENT postcard (address, sender?, message?)>  
<!ELEMENT address (name, postalcode, city, country)>  
<!ATTLIST address nick CDATA #IMPLIED>  
<!ATTLIST sender nick CDATA #IMPLIED>  
<!ATTLIST attachment type CDATA #REQUIRED>  
<!ATTLIST postcard scanimage CDATA #IMPLIED>  
<!ATTLIST sender name CDATA #REQUIRED>
```

A Possible XML for the given DTD

```
<mails>  
  <email>
```

```
<address nick="Alice">alice@usa.com</address>
<sender nick="Bob">bob@jp.com</sender>
<message>Hash code</message>
<attachment type="text/doc"></attachment>
</email>
<postcard scanimage="kep.jpg">
  <address>
    <name>John Doe</name>
    <postalcode>1234</postalcode>
    <city>Miskolc</city>
    <country>Hungary</country>
  </address>
  <sender name="Bob Cat" />
  <message>Happy Name Day</message>
</postcard>
</mails>
```

This provides an example of how attributes, including mandatory, optional, and default values, can be defined in a DTD and represented in an XML document.

From:
<https://edu.iit.uni-miskolc.hu/> - **Institute of Information Science - University of Miskolc**

Permanent link:
<https://edu.iit.uni-miskolc.hu/tanszek:oktatas:techcomm:dtd?rev=1764662267>

Last update: **2025/12/02 07:57**

