Week 6 Lecture: DTDs - Part 2

Screencast of Recorded Lecture (2022):

- https://sjeccd-edu.zoom.us/rec/play/M_9lfRzs8syAogpK44zNmjaF2bYDTUTnLbhKSVEvPnrQFC3nkWVhwBs8JQ-IesSWOjWu0waFnr7xizEF.Map6XCjwc9AU_EWT (https://sjeccd-edu.zoom.us/rec/play/M_9lfRzs8syAogpK44zNmjaF2bYDTUTnLbhKSVEvPnrQFC3nkWVhwBs8JQ-IesSWOjWu0waFnr7xizEF.Map6XCjwc9AU_EWT)

Screencast of Recorded Lecture (2021):

- https://sjeccd-edu.zoom.us/rec/play/MwjGoZxhVeQaOQpzSP7snv8YOnaswmJlbcRn1pF1Q8Bb6SfPp1cf6l479G9ctUeDhZGeWdczQCurMbaM.qJ6nBhx-64BQY_Lo (https://sjeccd-edu.zoom.us/rec/play/MwjGoZxhVeQaOQpzSP7snv8YOnaswmJlbcRn1pF1Q8Bb6SfPp1cf6l479G9ctUeDhZGeWdczQCurMbaM.qJ6nBhx-64BQY_Lo)

OK, Let's continue on with Document Type Definitions (DTD).

Let's start with:

Attributes

Attributes are declared with an ATTLIST declaration which has the following syntax:

```
<!ATTLIST  element-name  attribute-name attribute-type  default-value>
```

Here's an example:

> In the DTD:
>
> ```
> <!ATTLIST  payment  type   CDATA "check">
> ```
>
> In the XML data:
>
> ```
> <payment  type="check"  />
> ```

The **attribute-type** is any of the following values:

- CDATA	The value is character data, any legal XML string
- ENTITY	The value is an entity

- ENTITIES	The value is a list of entities
- ID	The value is a unique id
- IDREF	The value is the id of another element
- IDREFS	The value is a list of other values
- NMTOKEN	The value is a valid XML name
- NMTOKENS	The value is a list of valid XML names
- NOTATION	The value is a name of a notation
- xml:	The value is a predefined xml value

The **default-value** can have the following values:

- value	The default value of the attribute
- #REQUIRED	The attribute value must be included in the element
- #IMPLIED	The attribute does not have to be included
- #FIXED value	The attribute value is fixed

Here's a short example to show how it all goes together:

Suppose you have a very short XML document. This XML document shows that Customer # CA4567 bought 3 items (widget1, widget2, and widget3) on October 17, 2020, and had an order number of OCT123:

```
<order date="2020/10/17" ordernumber="OCT123" customerid="CA4567">
    <items>
        <widget1 />  <widget2 />  <widget3 />
    </items>
</order>
```

Notice that there are three attributes of the order element; for business purposes, each of these is essential. These are declared as a list of attributes associated with the order element.

The ATTLIST keyword implies an attribute list, but you can actually define as few as one attribute.

The DTD for this XML file might look like this:

```
<!ELEMENT order (items)>
<!ELEMENT items (widget1, widget2, widget3)>
<!ELEMENT widget1 EMPTY>
<!ELEMENT widget2 EMPTY>
<!ELEMENT widget3 EMPTY>
<!ATTLIST order
customerid CDATA #REQUIRED
date CDATA #REQUIRED
ordernumber CDATA #REQUIRED
>
```

- This DTD is stating that there will be an element "order".

- The element "order" contains an element "items" inside it,
- The element "items" contains 3 elements inside it -- "widget1", "widget2", and "widget3"
- widget1, widget2, and widget 3 are empty elements
- Element "order" contains attributes. Those attributes are "customerid", "date", and "ordernumber", and all 3 of these contain character data and are required attributes.

Here's a few more examples of attribute usage:

Specifying a Default attribute

If this is in the DTD:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

Then this is valid XML

```
<square width="123" />
```

In the above example, the "square" element is defined to be an empty element with a "width" attribute of type CDATA. If no width is specified, it has a default value of 0.

Using #IMPLIED

Syntax:

```
<!ATTLIST  element-name   attribute-name   attribute-type  #IMPLIED>
```

Example:

> If this is in the DTD:
>
> ```
> <!ATTLIST contact fax CDATA #IMPLIED>
> ```
>
> Then this is valid XML
>
> ```
> <contact fax="123-456-7890" />
> ```
>
> And this is also valid XML
>
> ```
> <contact />
> ```

You can use the #IMPLIED keyword when you don't want to force the author to include an attribute, and you don't specify a default value.

Using #REQUIRED

Syntax:

```
<!ATTLIST  element-name   attribute-name attribute-type  #REQUIRED>
```

Example:

> If this is in the DTD
>
> > ```
> > <!ATTLIST person number CDATA #REQUIRED>
> > ```
>
> Then this is valid XML because it contains the attribute **number**
>
> > ```
> > <person number="1234"  />
> > ```
>
> And these would be invalid XML because they don't have the attribute **number**
>
> > ```
> > <person />
> >
> > <person id="1234" />
> > ```

You use the #REQUIRED keyword if you want to force the attribute to be present.

Using #FIXED

Syntax

```
<!ATTLIST  element-name   attribute-name attribute-type  #FIXED   "value">
```

Example

> If this is in the DTD
>
> ```
> <!ATTLIST sender company CDATA #FIXED "Microsoft">
> ```
>
> Then this is valid XML
>
> ```
> <sender company="Microsoft" />
> ```
>
> But this would be invalid XML
>
> ```
> <sender company="Oracle" />
> ```

You can use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If the author of the XML document includes another value, the XML parser will return an error.

Entities

Entities are variables used to define shortcuts to common text. Entities are written starting with an ampersand and ending with a semicolon. There are a number of pre-defined entities, like &gt; for the > symbol.

- Entities references are references to entities
- Entities can be declared internal, or external

Internal Entity Declaration

Syntax

```
<!ENTITY  entity-name   "entity-value">
```

DTD example:

```
<!ENTITY writer "Joe Smith">
<!ENTITY copyright "Copyright Computing">
```

XML example:

```
<author>&writer; &copyright;</author>
```

This would parse as:

```
<author>Joe Smith Copyright Computing</author>
```

External Entity Declaration

Syntax

```
<!ENTITY  entity-name   SYSTEM  "URI/URL">
```

DTD example:

```
<!ENTITY writer SYSTEM "http://www.Computing.com/entities.xml">
<!ENTITY copyright SYSTEM "http://www.Computing.com/entities.dtd">
```

XML example:

```
<author>&writer; &copyright;</author>
```

This would parse as:

```
<author>http://www.Computing.com/entities.xml http://www.Computing.com/entities.dtd</author>
```

Here is an example of a TV Schedule DTD

```
<!DOCTYPE TVSCHEDULE [
<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY | PROGRAMSLOT+))+>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATINGS CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

The task of developing a DTD can be as simple or as difficult as you make it. It all depends on what you want to do with the information you intend to model with the DTD, and what you intend to do with the information once it has been marked up.

An easy, quick, and simple method of creating an DTD is to work backwards from the XML document.

Here's a simple example of an XML file used to generate a web page:

```
<?xml version="1.0"?>
<page>
<head>
  <title>My Home Page</title>
</head>
<body>
<title>Welcome to My Home Page</title>
<para>Sorry -- this  page is still under construction.</para>
</body>
</page>
```

Given the above example, we can sketch out the rough hierarchy of the elements, as follows:

- <page>
  - <head>
    - <title>
  - <body>
    - <title>
    - <para>

If we now transpose this into DTD syntax, we get the following:

```
<!DOCTYPE page [
   <!ELEMENT page (head, body)>
   <!ELEMENT head (title)>
   <!ELEMENT body (title, para)>
 ]>
```

All we have said is:

- The **page** is the root element.
- The **page** element consists of a **head** followed by a **body**.
- A **head** element contains a **title** element.
- A **body** element contains a **title** element followed by a **para** element.

This is the hierarchy of the elements. All we now need to add is occurrence indicators to specify any restrictions to the order of the elements along with how many times they can occur (if any).

The following will accomplish this:

```
<!ELEMENT page (head, body)>
<!ELEMENT head (title)>
<!ELEMENT body (title?, para)+>
```

The above will ensure that there is a **title** inside a **head**, and at least one **para** inside the **body**.

To complete the DTD all we have to do is fill in those elements that contain the actual text or at least don't contain any more elements.

This should finish it off:

```
<!ELEMENT title (#PCDATA)>
<!ELEMENT para (#PCDATA)>
```

Again, if it's an internal DTD, we put the DTD inside of the XML; if it's an external DTD, we "link" to the DTD.

So, recalling the "Star Wars" XML + DTD from last lecture, let's add an attribute called "ranking", and have as its content character data (numbers, in this case).

Given that, here are possible internal and external examples:

Internal Star Wars DTD Example

starwars1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE starwars [
<!ELEMENT starwars (movie*)>
<!ELEMENT movie (title, episode, year, director)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT episode (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT director (firstname,lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ATTLIST movie ranking CDATA #REQUIRED>
]>

<starwars>
 <movie ranking="2">
   <title>Star Wars: A New Hope</title>
   <episode>4</episode>
   <year>1977</year>
   <director>
     <firstname>George</firstname>
     <lastname>Lucas</lastname>
   </director>
 </movie>
 <movie ranking="1">
   <title>Empire Strikes Back</title>
   <episode>5</episode>
   <year>1980</year>
   <director>
     <firstname>Irvin</firstname>
     <lastname>Kershner</lastname>
   </director>
 </movie>
 <movie ranking="3">
   <title>Return of the Jedi</title>
   <episode>6</episode>
   <year>1983</year>
   <director>
     <firstname>Richard</firstname>
     <lastname>Marquand</lastname>
   </director>
 </movie>
</starwars>
```

External Star Wars DTD example

starwars2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE starwars SYSTEM "starwars.dtd">
<starwars>
 <movie ranking="2">
   <title>Star Wars: A New Hope</title>
   <episode>4</episode>
   <year>1977</year>
   <director>
     <firstname>George</firstname>
     <lastname>Lucas</lastname>
   </director>
 </movie>
 <movie ranking="1">
   <title>Empire Strikes Back</title>
   <episode>5</episode>
   <year>1980</year>
   <director>
     <firstname>Irvin</firstname>
     <lastname>Kershner</lastname>
   </director>
 </movie>
 <movie ranking="3">
   <title>Return of the Jedi</title>
   <episode>6</episode>
   <year>1983</year>
   <director>
     <firstname>Richard</firstname>
     <lastname>Marquand</lastname>
   </director>
 </movie>
</starwars>
```

starwars.dtd

```
<!ELEMENT starwars (movie*)>
<!ELEMENT movie (title, episode, year, director)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT episode (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT director (firstname,lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ATTLIST movie ranking CDATA #REQUIRED>
```

This last line indicates that the **movie** element has an attribute called **ranking**, and that the content of the value of **ranking** will be non-parsed character data, and that the value of the **ranking** must be included in the XML.