ỦY BAN NHÂN DÂN TỈNH LÂM ĐỒNG

**TRƯỜNG CAO ĐẲNG NGHỀ ĐÀ LẠT**

# GIÁO TRÌNH

## MÔN HỌC/ MÔ ĐUN: NGÔN NGỮ XML

## NGÀNH/ NGHỀ: THIẾT KẾ TRANG WEB

## TRÌNH ĐỘ: CAO ĐẲNG

*Ban hành kèm theo Quyết định số: 1157/QĐ-CĐNĐL ngày 11 tháng 12 năm 2019 của Hiệu trưởng Trường Cao đẳng Nghề Đà Lạt*

**(LƯU HÀNH NỘI BỘ)**

**Lâm Đồng, năm 2019**

## TUYÊN BỐ BẢN QUYỀN

# LỜI GIỚI THIỆU

*Vài nét về xuất xứ giáo trình:*

Giáo trình này được viết theo căn cứ Thông tư số 03/2017/TT-BLĐTBXH ngày 01 tháng 03 năm 2017 của Bộ Lao động – Thương binh và Xã hội về việc Quy định về quy trình xây dựng, thẩm định và ban hành chương trình; tổ chức biên soạn, lựa chọn, thẩm định giáo trình đào tạo trình độ trung cấp, trình độ cao đẳng.

*Quá trình biên soạn:*

Giáo trình này được biên soạn có sự tham gia tích cực của các giáo viên có kinh nghiệm, cùng với những ý kiến đóng góp quý báu của các chuyên gia về lĩnh vực công nghệ thông tin.

*Mối quan hệ của tài liệu với chương trình, mô đun/môn học:*

Căn cứ vào chương trình đào tạo nghề Thiết kế trang web, giáo trình giúp cung cấp cho người học những kiến thức cơ bản về ngôn ngữ XML cũng như kỹ năng xây dựng dữ liệu bằng ngôn ngữ XML.

Để học được mô đun này người học cần có kiến thức cơ bản về ngôn ngữ HTML và cơ sở dữ liệu.

*Cấu trúc chung của giáo trình này bao gồm 5 chương:*

Chương 1. XML (eXtensible Markup Language)

Chương 2. DTD (Document Type Definition)

Chương 3. Xpath (XML Path Language)

Chương 4. XSL (eXtensible style sheet)

Chương 5. XLink và Xpointer

*Lời cảm ơn*

Giáo trình được biên soạn trên cơ sở các văn bản quy định của Nhà nước và tham khảo nhiều tài liệu liên quan có giá trị. Song chắc hẳn quá trình biên soạn không tránh khỏi những thiếu sót nhất định. Ban biên soạn mong muốn và thực sự cảm ơn những ý kiến nhận xét, đánh giá của các chuyên gia, các thầy cô đóng góp cho việc chỉnh sửa để giáo trình ngày một hoàn thiện hơn.

*Lâm Đồng, ngày 10 tháng 12 năm 2019*

Tham gia biên soạn

1. Phạm Đình Nam
2. Ngô Thiên Hoàng
3. Nguyễn Quỳnh Nguyên
4. Phan Ngọc Bảo

# MỤC LỤC

# Chương mở đầu

Trong thời đại Công nghệ Thông tin hiện nay XML (**eXtensible Markup Language**) chiếm vị trí số một và rất quan trọng trong việc chuyển tải, trao đổi dữ liệu và liên lạc giữa các ứng dụng. Điều này càng được khẳn định khi trong các hệ điều hành từ WindowsXP trở đi, bên trong nó chứa đầy XML. Hơn nữa khi bộ .Net ra đời thì càng làm cho XML trở nên thịnh hành.

Sử dụng kỹ thuật XML không chỉ có tập đoàn Microsoft mà ngay cả Sun, IBM, Oracles điều hỗ trợ XML và  dùng nó trong các ứng dụng.

## 1 XML là gì?

XML là ngôn ngữ xây dựng cấu trúc tài liệu văn bản, dựa theo chuẩn SGML (Standard Generalized Markup Language: siêu ngôn ngữ có khả năng sinh ngôn ngữ khác). SGML được phát triển cho việc định cấu trúc và nội dung tài liệu điện tử, do tổ chức ISO (International Organization for Standards) chuẩn hoá năm 1986.

SGML là do IBM đưa ra, song không thể không kể đến những đóng góp của các công ty khác. XML được W3C (World Wide Web Consortium: tổ chức độc lập định ra tiêu chuẩn cho trình duyệt Web, máy chủ và ngôn ngữ) phát triển, nhưng đặc tả XML lại do Netscape, Microsoft và các thành viên của dự án Text Encoding Initiative (TEI) xây dựng. Tổ chức W3C XML Special Interest Group có đại diện từ hơn 100 công ty cùng nhiều chuyên gia được mời khác.

Lý do ra đời của XML vì SGML rất rắc rối, và HTML có nhiều giới hạn nên năm 1996 tổ chức W3C thiết kế XML. XML version 1.0 được định nghĩa trong hồ sơ **February 1998 W3C Recommendation.**

Điểm quan trọng của kỹ thuật XML là nó không thuộc riêng về một công ty nào, nó là một sản phẩm mà trí tuệ của nó thuộc về cả thế giới, nó là một tiêu chuẩn được mọi người công nhận vì được soạn ra bởi World Wide Web Consortium - W3C (một ban soạn thảo với sự hiện diện của tất cả các chuyên gia Tin học) và những ý kiến đóng góp bằng cách trao đổi qua Email.

Bản thân của XML rất là đơn giản, nhưng các công cụ chuẩn được định ra để làm việc với XML như **Document Object Model - DOM, XPath, XSL**, v.v.. thì rất hữu hiệu, và chính các chuẩn này được phát triển không ngừng.

XML cũng giống như HTML đều là ngôn ngữ đánh dấu, nhưng điều cần nói ở đây là sự ra đời của XML để khắc phục cho một số yếu kém của HTML. HTML và XML đều sử dụng các tag nhưng các tag của HTML là một bộ dữ liệu tag được xây dựng và định nghĩa trước, tức là người lập trình phải tuân thủ theo các thẻ đã định nghĩa của HTML, hiện HTML có khoản hơn 400 tag, để nhớ hết 400 tag này cũng không có gì khó khăn đối với người lập trình web chuyên nghiệp nhưng thật khó đối với những người không chuyên. Hơn nữa các tag của HTML không nói lên được mô tả dữ liệu trong đó. Nhưng đối với XML thì hoàn toàn khác bởi vì tag trong XML là do người lập trình định nghĩa và mỗi tag là một mô tả dữ liệu mà người lập trình muốn truyền đạt.

## 2 Các nội dung sẽ trình bày

Khi XML ra đời thì có hàng loạt các ngôn ngữ chuẩn được đưa ra để làm việc với XML, nhưng trong tài liệu này tôi chỉ xin được trình bày các phần chính sau:

1. Ngôn ngữ XML(eXtensible Markup Language )

   Ngôn ngữ xây dựng cấu trúc tài liệu văn bản, dựa theo chuẩn SGML (Standard Generalized Markup Language

2. Ngôn ngữ DTD (Document Type Definition)

Ngôn ngữ dùng để định nghĩa kiểu dữ liệu cho các phần tử trong tài liệu XML

3. Ngôn ngữ Xpath (XML Path Language) Ngôn ngữ dung để duyệt tài liệu XML
4. XSL (eXtensible style sheet)

   Ngôn ngữ dùng để chuyển đổi tài liệu XML thành một định dạng khác
5. Ngôn ngữ Xlink(XML Link Language) và Xpointer(XML Pointer Language)

Ngôn ngữ lien kết và định vị tài liệu

# Chương 1
# XML (eXtensible Markup Language)

## 1 Phần lý thuyết

### *1.1 Các quy tắc cần lưu ý*

Để viết được một trang XML cũng rất đơn giãn, chúng ta chỉ cần tuân thủ những quy tắc sau:

- Phải có một Phần tử gốc duy nhất, nó chứa tất cả các Phần tử khác trong tài liệu.

```
<Catalog>
  <Product ProductID="1">Chair</Product>
  <Product ProductID="2">Desk</Product>
</Catalog>
```

- Mỗi Tag mở  phải có một Tag đóng giống như nó.

```
1   <Order>
2     <OrderDate>2002-6-14</OrderDate>
3 4   <Customer>Helen Mooney</Customer>
5 6   <Item>
7       <ProductID>2</ProductID>
8 9     <Quantity>1</Quantity>
10    <Item>
11      <ProductID>4</ProductID>
        <Quantity>3</Quantity>
      </Item>
    </Order>
```

Ví dụ trên không thỏa qui tắc 2 vì thiếu Tag đóng </Item>, ta cần thêm Tag đóng </item> vào sau dòng thứ 6

- Trong một cặp Tag đóng và mở phải được đánh vần như nhau, có nghĩa là các cặp ký tự của cặp Tag  đóng mở này phải giống nhau hoàn toàn.

```
1   <Order>
2     <OrderDate>2001-01-01</Orderdate>
3     <Customer>Graeme Malcolm</Customer>
4   </Order>
```

Ví dụ này không thỏa quy tắc 3 vì Tag mở <OrderDate> và Tag đóng </Orderdate> đánh vần không giống nhau, ta phải sửa Tag đóng </Orderdate> thành </OrderDate> hoặc sửa thẻ <OrderDate> thành <Orderdate>

- Mỗi  Phần tử con phải nằm trọn bên trong Phần tử cha của nó.

```
1  <Catalog>
2    <Category CategoryName="Beverages">
3      <Product ProductID="1">
4        Coca-Cola
5      </Category>
6    </Product>
7  </Catalog>
```

Ví dụ này không thỏa quy tắc 4 vì Tag đóng </Product> đặt không đúng vị trí, ta cần đổi vị trí của dòng 5 cho dòng 6

- Giá trị của thuộc tính phải được đặt trong cặp dấu nháy kép hoặc cặp dấu nháy đơn.

```
1  <Catalog>
2    <Product ProductID=1>Chair</Product>
3    <Product ProductID='2">Desk</Product>
4  </Catalog>
```

Ví dụ này không đúng quy tắc 5 vì giá trị của thuộc tính ProductID ở dòng 3 đặt trong cặp dấu nháy không đúng, ta cần sửa lại ProductID='2" thành ProductID=”2" hoặc ProductID='2’

## *1.2 Chỉ thị xử lý (Processing Instructions) và lời chú thích (Comments)*

Chúng ta thường thấy dòng lệnh

<?xml version="1.0" encoding="utf-8"  standalone="yes"?>

nằm ở đầu file XML. Đây chính là chỉ thị xử lý, chỉ thị xử lý được đặt trong cặp Tag <? và ?>. Nó cho biết phiên bản đặc tả XML mà bộ phân tích cần làm theo, ngoài ra nó cho phép người lập trình cho biết dữ liệu trong XML dùng encoding nào, còn thuộc tính standalone sẽ cho biết  tài liệu XML có cần đến một tài liệu khác không (có hai giá trị cho thuộc tính này đó là “yes” nếu không cần đến một tài liệu khác và “no” nếu cần).

```
<?xml version="1.0" encoding="utf-8"  standalone="yes"?>
    <Order>
      <OrderDate>2002-6-14</OrderDate>
      <Customer>Helen Mooney</Customer>
      <Item>
        <ProductID>1</ProductID>
        <Quantity>2</Quantity>
      </Item>
      <Item>
        <ProductID>4</ProductID>
        <Quantity>1</Quantity>
      </Item>
</Order>
```

Trong mỗi tài liệu XML có thể có hoặc không có phần này

Để file XML trở nên dễ hiểu và dể chỉnh sửa sau này thì các dòng chú thích là không thể thiếu, các dòng chú thích được đặt trong cặp tags <!--  và -->.

## *1.3 Không gian tên (namespace).*

XML cho phép chúng ta tự do định nghĩa các thẻ, như nó cho chúng ta dùng cùng một tên nhưng lại nói đến nhiều loại dữ liệu khác nhau trong cùng một tài liệu XML. Xem ví dụ sau:

```
<?xml version="1.0"?>
<BookOrder OrderNo="1234">
  <OrderDate>2001-01-01</OrderDate>
  <Customer>
    <Title>Mr.</Title>
    <FirstName>Graeme</FirstName>
<LastName>Malcolm</LastName>
  </Customer>
  <Book>
    <Title>Treasure Island</Title>
    <Author>Robert Louis Stevenson</Author>
</Book>

</BookOrder>
```

Ta thấy trong ví dụ trên có phần tử Title nói đến hai loại dữ liệu khác nhau, một nói về tên tác giả một nói về tiêu đề sách, điều này làm cho ta nhầm lẫn giữa hai loại dữ liệu. Hơn thế nữa nếu tài liệu của chúng ta được sử dụng chỉ cho một mục đích riêng rẻ thì không có vấn đề gì nhưng khi tài liệu của chúng ta kết hợp với một tài liệu khác khác thì tài liệu kết hợp này sẽ có vấn đề vì chúng ta đâu chắc chắn rằng tài liệu khác mà chúng ta muốn kết hợp không có sử dụng thẻ trùng với thẻ của chúng ta định nghĩa hay không.

Ví dụ như khi chúng ta tích hợp tài liệu XML của chúng ta với ứng dụng khác như VML hay MathML mà chẳng may giữa các tài liệu này có cùng định nghĩa thẻ NAME chẳng hạn. Lúc này trình phân tích sẽ không biết nên hiểu thẻ NAME của tài liệu của bạn hay của VML hay của MathML.

Vì vậy chúng ta cần phải khai báo không gian tên để khắc phục điều này.

### *1.3.1 Khai báo không gian tên (namespace)*

Để khai báo một không gian tên ta chỉ cần đưa thêm thuộc tính **xmlns:prefix** vào bên trong phần tử gốc, **prefix** là tên của không gian tên, mỗi không gian tên cần mang một định danh duy nhất. Một không gian tên có thể là một địa chỉ internet hoặc một địa chỉ nào đó miễn là địa chỉ này phải duy nhất. Ví dụ sau đây sẽ tạo ra một không gian tên **hs** và áp dụng cho tất cả các phần tử con:

```
<?xml version="1.0"?>
<BookOrder  xmlns:hs="http://www.northwindtraders.com/customer">
  <hs:Customer >
    <hs:Title>Mr.</Title>
    <hs:FirstName>Graeme</FirstName>
<hs:LastName>Malcolm</LastName>    </hs:Customer>

</BookOrder>
```

### *1.3.2 Không gian tên mặc định (namespace default)*

Nếu tài liệu của chúng ta các phần tử chỉ sử dụng duy nhất một không gian tên thì chúng ta có thể khai báo không gian tên mặc định cho các phần tử con của một phần tử cha bằng cách chỉ ghi thuộc tính **xmlns** và bỏ đi **prefix**

```
<?xml version="1.0"?>
<BookOrder  xmlns ="http://www.northwindtraders.com/customer">
  < Customer >
    < Title>Mr.</Title>
    < FirstName>Graeme</FirstName>
< LastName>Malcolm</LastName>    </
Customer>
</BookOrder>
```

Chúng ta xem tiếp ví dụ sau:

```
<?xml version="1.0"?>
<BookOrder >
  <OrderDate>2001-01-01</OrderDate>
  <Customer xmlns="http://www.northwindtraders.com/customer">
<Title>Mr.</Title>
    <FirstName>Graeme</FirstName>
<LastName>Malcolm</LastName>
  </Customer>
  <Book xmlns="http://www.northwindtraders.com/book">
    <Title>Treasure Island</Title>
    <Author>Robert Louis Stevenson</Author>
</Book>
</BookOrder>
```

Ví dụ trên chúng ta thấy có hai không gian tên mặc định, một cho phần tử **Customer** và một cho phần tử **Book**. Nhưng điều đáng nói ở đây là nếu như trong tài liệu có nhiều **Customer** và nhiều **Book** thì chúng ta không thể viết đi viết lại không gian tên mãi được, rất mất thời gian.

Cách giải quyết tốt nhất là ta khai báo các không gian tên này ngay ở đầu tài liệu và mỗi không gian tên được phân biệt bởi các định danh.

```
<?xml version="1.0"?>
<BookOrder  xmlns="http://www.northwindtraders.com/order"
                xmlns:cus="http://www.northwindtraders.com/customer"
                xmlns:bok      ="http://www.northwindtraders.com/book"
>
<OrderDate>2001-01-01</OrderDate>
  <cus:Customer >
    <cus:Title>Mr.</cus:Title>
    <cus:FirstName>Graeme</cus:FirstName>
<cus:LastName>Malcolm</cus:LastName>
  </cus:Customer>
  <bok:Book >
    <bok:Title>Treasure Island</bok:Title>
    <bok:Author>Robert Louis Stevenson</bok:Author>
</bok:Book>
</BookOrder>
```

Ví dụ trên dùng 3 không gian tên, một không gian tên mặc định và hai không gian tên có định danh là **cus** và **bok**. Trong ví dụ trên những phần tử không có định danh của không gian tên đi trước thì được hiểu là sử dụng không gian tên mặc định http://www.northwindtraders.com/order, như phần tử **<OrderDate>2001-01-01</OrderDate>**

## *1.4 CDATA*

Đoạn dữ liệu của CDATA là đoạn dữ liệu nằm giữa <![CDATA [ và ]]>. Những đoạn dữ liệu nằm trong CDATA khi đi qua trình phân tích sẽ được giữ nguyên

như ban đầu, tức là khi gặp CDATA thì trình phân tích sẽ bỏ qua. Điều này rất cần thiết khi chúng ta viết những đoạn mã script trong tài liệu.

```
<script language=”javascript”>
<![CDATA[ function
mag(){
   alert(“This is CDATA! ”);
}
]]
</script>
```

### *1.5 Thực thể định nghĩa sẵn trong XML*

Trong ngôn ngữ định dạng XML có sử dụng một số ký tự định dạng đặc biệt: <, >, ‘, “, &. Vì vậy để giúp cho chúng ta thể hiện tài liệu đúng theo nguyên mẫu bằng cách định nghĩa 5 thực thể này như sau:

| **Thực thể** | **Mô tả** |
|---|---|
| &apos; | Tương đương với dấu nháy đơn (‘) |
| &amp; | Tương đương với dấu & |
| &gt; | Tương đương với dấu > |
| &lt; | Tương đương với dấu < |
| &quot; | Tương đương với dấu nháy kép (“) |

Ví dụ:

```
<?xml version="1.0"?>
<LINK-TO>
   &lt; a href=&quot; index.txt &quot; &gt; OPEN FILE INDEX.TXT &lt; /a &gt;
</LINK-TO>
```

(< a href=" index.txt " > OPEN FILE INDEX.TXT < /a > tương đương với <a href=index.txt> OPEN FILE INDEX.TXT </a>)

Để hiểu rõ hơn về thực thể là gì, chúng ta sẽ tìm hiểu trong chương 2.

## 2 Phần ví dụ

# Chương 2
# DTD (Document Type Definition)

## 1 DTD là gì?

DTD (Document Type Definition) là kiểu tài liệu dùng để định nghĩa kiểu dữ liệu cho các phần tử trong tài liệu XML. Khi chúng ta định nghĩa các phần tử trong XML là tùy thích, miễn sao cho nó hợp quy tắc của tài liệu XML. Tuy nhiên để tường minh hơn thì ta nên định nghĩa kiểu dữ liệu cho từng phần tử trong tài liệu XML.

Trong chương trước chúng ta đã học cách viết một tài liệu hợp khuôn dạng. Tuy nhiên một tài liệu XML được xem là hợp khuôn dạng và có giá trị khi toàn bộ các phần tử trong tài liệu được được định nghĩa kiểu dữ liệu mà nó chứa. Với cách định nghĩa kiểu tư liệu (DTD) khi chúng ta đọc một tài liệu XML nào thì chỉ cần đọc phần DTD thì chúng ta sẽ biết được cấu trúc của tài liệu XML.

Trước khi đi vào phần chi tiết về cách tạo một tài liệu DTD, chúng ta hãy xem ví dụ sau:

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

Xem ví dụ trên chúng ta thấy phần được bôi đen là phần DTD dùng để mô tả kiểu dữ liệu của tài liệu XML, phần có màu nhạt là các phần tử của tài liệu XML.

Phần DTD trong ví dụ này được hiểu như sau: Tài liệu XML có một phần tử gốc tên là note, phần tử gốc này có 3 phần tử con là from, heading, body và 3 phần tử con này có kiểu dữ liệu text.

## 2 Định nghĩa một tài liệu DTD

Để viết một tài liệu DTD cũng rất dễ, chỉ cần chúng ta tuân thủ đúng một số quy tắc của W3C là được. Đầu tiên chúng ta hãy tìm hiểu về các phần tử (emlement), thuộc tính, thực thể của DTD.

### *2.1 Phần tử <!DOCTYPE>*

Phần tử này có chức năng dùng để khai báo bắt đầu định nghĩa kiểu tư liệu DTD.

Định nghĩa kiểu tư liệu có 2 dạng, đó là DTD tham chiếu nội và DTD tham chiếu ngoại. DTD tham chiếu nội là DTD được định nghĩa ngay trong tài liệu XML còn

DTD tham chiếu ngoại là DTD được định nghĩa bên ngoài tài liệu XML. Bây giờ chúng ta sẽ tìm hiểu từng cú pháp một.

### *2.1.1 Định nghĩa DTD tham chiếu nội*

Để bắt đầu định nghĩa kiểu tư liệu DTD tham chiếu nội chúng ta dùng cú pháp sau:

**<!DOCTYPE root-element [DTD]**

Trong đó root-element là phần tử gốc của tài liệu XML, DTD là các định nghĩa cho các phần tử trong tài liệu XML.

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note body>
  <!ELEMENT body    (#PCDATA)>
]>
```

```
 <note>
    <body>Don't forget me this weekend</body>
</note>
```

### *2.1.2 Định nghĩa DTD tham chiếu ngoại*

Sử dụng định nghĩa DTD tham chiếu ngoại sẽ làm cho các ứng dụng XML của chúng ta trở nên dễ dàng chia sẽ và dùng chung với các ứng dụng khác. Có hai cách để chỉ định một DTD tham chiếu ngoại: Tham chiếu ngoại riêng và tham chiếu ngoại chung.

Những định nghĩa DTD tham chiếu ngoại riêng được sử dụng cho một nhóm người mang tính cá nhân, chúng không được dùng cho mục đích chung rộng lớn, mục đích phân phối. Còn những định nghĩa DTD tham chiếu ngoại chung sẽ mang tính cộng đồng hơn.

- Để định nghĩa một DTD tham chiếu ngoại riêng chúng ta dùng cú pháp sau:

  **<!DOCTYPE root-element  SYSTEM “filename”>**

Trong đó root-element là tên của phần tử gốc trong tài liệu XML, filename là tên file định nghĩa kiểu tư liệu DTD Ví dụ:

```
<?xml version="1.0"?>
 <!DOCTYPE note SYSTEM "note.dtd">
  <note>
     <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
 </note>
```

File note.dtd với nội dung như sau:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Địa chỉ chứa file DTD có thể một URL/URI.

```
<?xml version="1.0"?>
 <!DOCTYPE note SYSTEM "http://www.w3schools.com/dtd/note.dtd">
  <note>
     <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
 </note>
```

- Để định nghĩa một DTD tham chiếu ngoại chung chúng ta dùng cú pháp sau:

**<!DOCTYPE root-element PUBLIC “FPI” “URL”>**

Trong đó FPI (Formal Public Identifier) là một định danh chung hình thức, chúng ta cần tuân theo một số quy tắc áp dụng cho FPI sau:

- Trường đầu tiên của một FPI là xác định kết nối của DTD đến chuẩn hình thức. Đối với các DTD chúng ta tự định nghĩa thì trường này là một dấu chấm. Đối với các chuẩn hình thức trường này sẽ tự tham chiếu đến chuẩn của nó.
- Trường thứ hai là tên nhóm hay tên người chịu trách nhiệm bảo trì và nâng cấp các định nghĩa DTD và tên này phải mang tính duy nhất.
- Trường thứ ba chỉ định kiểu của tài liệu được mô tả, thường thì trường này kèm theo một số định danh duy nhất nào đó (chẳn hạn như version 1.0).
- Trương thứ ba chỉ định ngôn ngữ mà bạn định nghĩa DTD (ví dụ như ngôn ngữ Tiếng Anh - EN)
- Mỗi trường của FPI cách nhau bởi dấu //

Ví dụ

```
<?xml version="1.0"?>
  <!DOCTYPE note PUBLIC “.//w3schools//note XML version 1.0//EN”
 "http://www.w3schools.com/dtd/note.dtd">
   <note>
      <to>Tove</to>
       <from>Jani</from>
       <heading>Reminder</heading>
     <body>Don't forget me this weekend!</body>
  </note>
```

Còn URL là địa chỉ của file DTD.

## *2.2 Phần tử <!ELEMENT>*

Phần tử <!ELEMENT> dùng để định nghĩa kiểu dữ liệu cho một phần tử của một tài liệu XML. Chúng ta sử dụng theo cú pháp sau: <!ELEMENT element-name content_model> Trong đó:

- **element_name** là tên của phần tử mà ta muốn định nghĩa
- **content_model** là kiểu của phần tử này, có thể là EMPTY, ANY, #PCDATA, các phần tử con hay trộn lẫn nhiều thành phần Bây

giờ chúng ta tìm hiểu chi tiết hơn.

- Định nghĩa một phần tử rỗng

  **<!ELEMENT element_name EMPTY>**

  Ví dụ: <!EMLEMENT note EMPTY>

- Định nghĩa một phần tử có chứa nhiều kiểu dữ liệu

  **<!ELEMENT element_name ANY>**

  Ví dụ: <!ELEMENT note ANY>

- Định nghĩa một phần tử có kiểu văn bản

  **<!ELEMENT element_name (#PCDATA)>**

  Ví dụ: <!ELEMENT note (#PCDATA)>

- Định nghĩa một phần tử có chứa một phần tử con

  **<!ELEMENT element_name (child_element)>**

  Ví dụ: <!ELEMENT note (to)>

- Định nghĩa một phần tử có chứa nhiều hơn một phần tử con, cách thứ nhất là chúng ta có thể liệt kê tất cả các phần tử con đó và mỗi phần tử con cách nhau bởi dấu phẩy.

Ví dụ để khai báo phần tử note có 4 phần tử con là to, from, heading, body chúng ta viết như sau:

<!ELEMENT note (to, from, heading, body)>

Tất nhiên với cách viết như thế này thì không tối ưu, chúng ta có thể dùng cách viết thứ hai cho những phần tử có nhiều phần tử con bằng cách dùng ký tự đại diện.

Dưới đây là một số nguyên tắc sử dụng ký tự đại diện:

Giả sử chúng ta có phần tử ROOT, phần tử này có hai phần tử con là LIMB_A và LIMB_B, chúng ta có một số định nghĩa sau:

- <!ELEMENT ROOT (LIMB_A*)>

  Phần tử ROOT  không có hoặc có nhiều phần tử LIMB_A •

<!ELEMENT ROOT (LIMB_A+)>

Phần tử ROOT có một hoặc nhiều phần tử con LIMB_A

- <!ELEMENT ROOT (LIMB_A?)>

Phần tử ROOT không có hoặc có một phần tử con LIMB_A

- <!ELEMENT ROOT (LIMB_A, LIMB_B)>

  Phần tử ROOT có 2 phần tử con, đầu tiên là phần tử LIMB_A tiếp đến là LIMB_B

- <!ELEMENT ROOT (LIMB_A | LIMB_B)>

Phần tử ROOT có một phần tử con hoặc là LIMB_A hoặc là LIMB_B

- Định nghĩa một phần tử có chứa phần tử con hoặc chứa dữ liệu văn bản

<!ELEMENT LIMB_A (LIMB_A1| #PCDATA)>

## *2.3 Phần tử <!ATTLIST>*

Phần tử <!ATTLIST> dùng để định nghĩa kiểu tư liệu của các thuộc tính cho một phần tử trong tài liệu XML. Chúng ta dùng cú pháp sau:

**<!ATTLIST element-name attribute-name  attribute-type default-value>**
Trong đó: ○ **element-name** là tên của một phần tử cần định nghĩa thuộc tính ○ **attribute-name** là tên thuộc tính cần định nghĩa ○ **attribute-type** kiểu của thuộc tính. Có thể nhận một tong các giá trị sau:

| **Kiểu** | **Mô tả** |
|---|---|
| CDATA | Cho biết thuộc tính này chỉ có thể chứa kiểu dữ liệu ký tự |
| (en1\|en2\|..) | Danh sách các giá trị mà thuộc tính có thể được gán |
| ID | Cho biết thuộc tính này là một ID, tức là các giá trị của thuộc tính này không được trùng nhau và phải bắt đầu bởi một chữ cái |
| IDREF | Cho biết giá trị của thuộc tính này phải là một trong các giá trị của thuộc tính ID của các phần tử khác |
| IDREFS | Cho biết giá trị của thuộc tính này phải là các giá trị của các thuộc tính có kiểu ID |
| NMTOKEN | Cho biết giá trị của thuộc tính là các giá trị hợp với quy tắc đặt tên của phần tử của tài liệu XML |
| NMTOKENS | Cũng giống như NMTOKEN nhưng nó cho phép chứa nhiều NMTOKEN |
| ENTITY | Cho biết thuộc tính này nhận giá trị là một tên tham chiếu của thực thể |
| ENTITIES | Cho biết thuộc tính này nhận giá trị là các tên tham chiếu của thực thể và cách nhau bởi khoản trắng |
| NOTATION | (tôi chưa hiểu kiểu này) |
| xml: | (tôi chưa hiểu kiểu này) |

○ **default-value** thông tin về giá mặc định trị của thuộc tính này. Nó có thể nhận một trong các giá tị sau:

| **Giá trị** | **Mô tả** |
|---|---|

| | |
|---|---|
| value | value là một giá trị mặc định nào đó cho giá trị này (ví dụ “CNTT”) |
| #REQUIRED | Chỉ định là không có giá trị mặc định cho thuộc tính này, nhưng khi sử dụng là phải khởi tạo |
| #IMPLIED | Chỉ định là không có giá trị mặc định cho thuộc tính này, và thuộc tính này không cần dùng đến |
| #FIXED value | Chỉ định thuộc tính này chỉ mang duy nhất giá trị value này |

Chúng ta có thể định nghĩa một phần tử có nhiều thuộc tính theo cú pháp sau:

**<!ATTLIST element-name**
**attribute-name_1 attribute-type_1 default-value_1**
**attribute-name_2 attribute-type_2 default-value_2**
**...**
**attribute-name_n attribute-type_n default-value_n>**

(Xem ví dụ1) Ví dụ1:

Giả sử chúng ta có file att.dtd với nội dung sau:

```
<!ELEMENT attributes (#PCDATA)>
 <!ATTLIST attributes   aaa CDATA #REQUIRED  bbb CDATA #IMPLIED>
```

File XML chúng ta viết như sau:

```
<?xml version="1.0"?>
<!DOCTYPE attributes SYSTEM "att.dtd">
 <attributes aaa="#d1" bbb="*~*">Text</attributes>
```

Ví dụ2:

Giả sử chúng ta có file att.dtd với nội dung sau:

```
<!ELEMENT attributes (#PCDATA)>
    <!ATTLIST attributes   aaa CDATA #IMPLIED
```

```
                    bbb NMTOKEN #REQUIRED
                    ccc NMTOKENS #REQUIRED>
```

File XML chúng ta viết như sau:

```
<?xml version="1.0"?>
<!DOCTYPE attributes SYSTEM "att.dtd">
    <attributes aaa="#d1" bbb="a1:12" ccc=" 3.4 div   -4"/>
```

Nếu chúng ta viết như sau sẽ không hợp quy tắc vì kiểu NMTOKEN và NMTOKEN không chấp nhận ký tự # :

```
<?xml version="1.0"?>
<!DOCTYPE attributes SYSTEM "att.dtd">
 <attributes aaa="#d1" bbb="#d1" ccc="#d1"/>
```

Ví dụ3:

Giả sử chúng ta có file att.dtd với nội dung sau:

```
<!ELEMENT XXX (AAA+ , BBB+ , CCC+)>
   <!ELEMENT AAA (#PCDATA)>
  <!ELEMENT BBB (#PCDATA)>
   <!ELEMENT CCC (#PCDATA)>
     <!ATTLIST AAA  id ID #REQUIRED>
     <!ATTLIST BBB  code ID #IMPLIED
                    list NMTOKEN #IMPLIED>
     <!ATTLIST CCC X ID #REQUIRED
                    Y NMTOKEN #IMPLIED>
```

File XML chúng ta viết như sau:

```
<?xml version="1.0"?>
<!DOCTYPE XXX SYSTEM "att.dtd">
    <XXX>
      <AAA id="L12"/>
      <BBB code="QW" list="L12"/>
      <CCC X="x-0" Y="QW" />
      <CCC X="x-1" Y="QW" />
    </XXX>
```

Nếu chúng ta viết như sau sẽ không hợp quy tắc vì phần tử CCC có thuộc tính X có kiểu là ID nên phải là duy nhất.

```
<?xml version="1.0"?>
<!DOCTYPE XXX SYSTEM "att.dtd">
    <XXX>
```