Bài 2. XML và XSD

- Mục đích, yêu cầu: Cung cấp cho sinh viên kiến thức về XML và XSD, qua đó sinh viên có thể tổ chức được file lưu trữ XML cũng như xây dựng được đồ cấu trúc file XML.

- Hình thức tổ chức dạy học: Lý thuyết, trực tuyến + tự học

- Thời gian: Lý thuyết (trên lớp: 3; online: 3) Tự học, tự nghiên cứu: 12

- Nội dung chính:

XML và XSD

I. LÝ THUYẾT -TỔNG QUAN VỀ XML

1- XML là gì

- XML, hoặc Extensible Markup Language (ngôn ngữ đánh dấu mở rộng), là một ngôn ngữ đánh dấu mà bạn có thể sử dụng để tạo ra **thẻ riêng** của mình.
- Nó được tạo nên bởi Liên minh mạng toàn cầu nhằm khắc phục những hạn chế của HTML - ngôn ngữ đánh dấu siêu văn bản, là cơ sở của mọi trang Web.
- Do tạo được các thẻ riêng nên XML mềm dẻo hơn HTML rất nhiều, vì XML là các thẻ cung cấp thông tin, cho nên nó truyền thông và tìm kiếm nhanh hơn khi truy cập vào database

2 - Tại sao chúng ta cần XML?

- HTML là ngôn ngữ đánh dấu thành công nhất từ trước tới nay. Bạn có thể thấy dấu ấn của HTML đơn giản nhất trên bất cứ công cụ nào, từ thiết bị cầm tay tới máy chủ, thậm chí bạn còn có thể chuyển đổi đánh dấu HTML sang lời nói hoặc các định dạng khác với những công cụ chính xác. HTML thành công như thế, tại sao W3C lại tạo ra XML? Để trả lời cho câu hỏi này, hãy xem tài liệu dưới đây:

```
<p><b>Mrs. Mary McGoon</b>
<br>
1401 Main Street
<br>
Anytown, NC 34829</p>
```

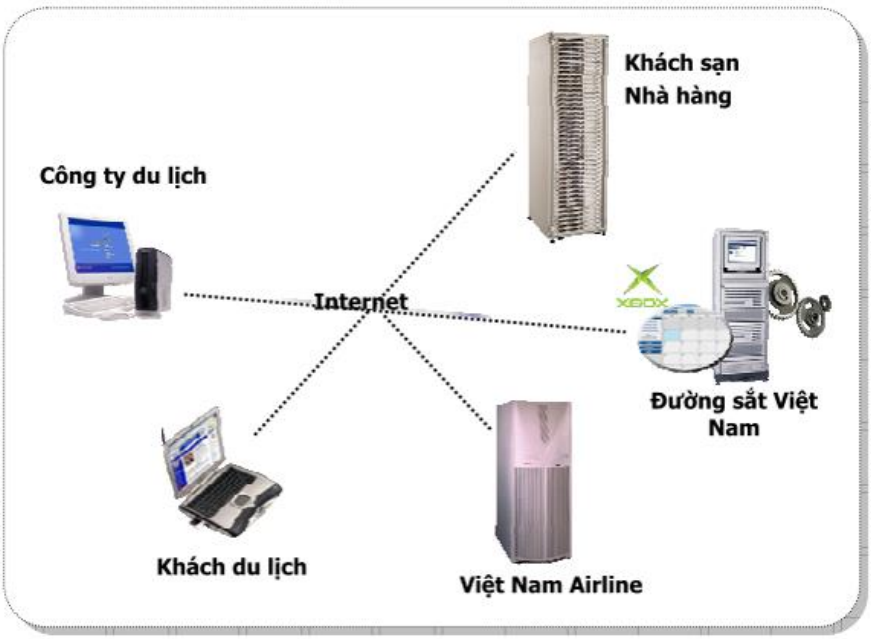
- Thẻ HTML này chỉ cho chúng ta biết thông tin nhưng máy tính không sử dụng được các thông tin này để làm gì cả
- Với XML, bạn có thể đưa ý nghĩa vào các thẻ trong văn bản. Quan trọng hơn, máy tính sẽ dễ dàng hơn trong việc xử lý thông tin. Bạn có thể rút được mã bưu chính ra

từ văn bản này đơn giản là bằng cách bao bọc nó bởi <postal-code> và </postal-code>

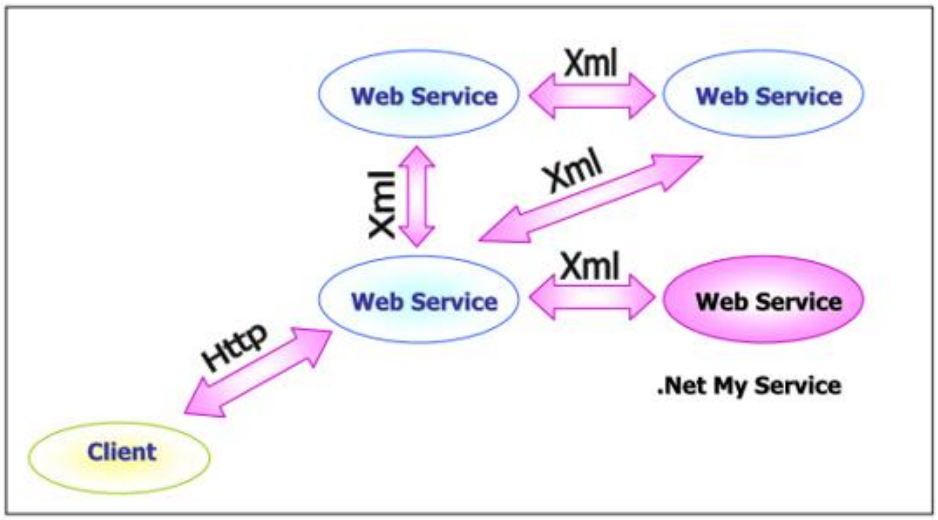
```
<?xml version="1.0" encoding="utf-8"?>
<address>
  <name>
    <title>Mrs.</title>
    <first-name>
      Mary
    </first-name>
    <last-name>
      McGoon
    </last-name>
  </name>
  <street>
    1401 Main Street
  </street>
  <city>Anytown</city>
  <state>NC</state>
  <postal-code>
    34829
  </postal-code>
</address>
```

3 - Ứng dụng XML

- **Xml là gì**? XML là ngôn ngữ đánh dấu, được dùng để miêu tả dữ liệu. Các thẻ (tag) trong XML chưa xác định trước. Người dùng tự định nghĩa trong quá trình tạo tài liệu XML.
- Trong thực tế XML được sử dụng để đóng gói và trao đổi dữ liệu giữa các hệ thống.



Hình 1: Mối liên kết giữa các hệ thống



Hình 2: Dữ liệu trao đổi giữa các hệ thống

- Khi có sự trao đổi dữ liệu giữa các hệ thống khác nhau thì dữ liệu đó được tổ chức dưới dạng XML. Hệ thống quản lý của **Nhà hàng** muốn lấy thông tin của khách du lịch từ hệ thống của **Công ty du lịch** thì giữa các hệ thống cần phải thực hiện các bước sau:
    - Bước 1: Giữa các hệ thống phải thống nhất cấu trúc của tài liệu XML
    - Bước 2: Công ty du lịch sẽ trích xuất dữ liệu từ hệ thống của mình, sau đó đóng gói dữ liệu dưới dạng XML theo cấu trúc đã thoả thuận ở bước 1.
    - Bước 3: Hệ thống phần mềm của nhà hàng sẽ tiến hành phân tích và trích xuất dữ liệu từ tài liệu XML nhận được từ hệ thống của công ty du lịch.

4 - Tạo 1 tài liệu XML

- Khái báo XML – cú pháp

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

- Tạo phần tử gốc – mỗi tài liệu XML phải có 1 phần tử gốc

```
<?xml version="1.0" encoding="utf-8" ?>
<SACH>
    ....
</SACH>
```

- Tạo mã XML – Mã XML bao gồm dữ liệu nằm giữa 2 thẻ Mở và Đóng
    - Thẻ Mở và đóng cùng 1 khuôn dạng
    - Phân biệt chữ hoa và thường
    - Thẻ đóng phải chính xác với thẻ mở

- Các thẻ phải lồng nhau hợp lý
- Thẻ không có dấu space
- Thẻ có độ dài hợp lý

```
<?xml version="1.0" encoding="utf-8" ?>
<THUVIEN>
   <SACH>
     <MASACH>001</MASACH>
     <TENSACH>Cơ sở dữ liệu</TENSACH>
   </SACH>
   <SACH>
     <MASACH>000</MASACH>
     <TENSACH>xml</TENSACH>
   </SACH>
</THUVIEN>
```

5 - Các thuộc tính của thẻ

- Các thẻ có thể chỉ rõ tên các thuộc tính hỗ trợ. Các thuộc tính không thể trùng nhau trong 1 thẻ.
- Tên thuộc tính và giá trị gán cho nó phải thông qua dấu '='
- Tất cả giá trị thuộc tính đề có kiểu là xâu => đều trong dấu " "

<OTO MODEL="Vios" MAU="Trắng">

<SINHVIEN MAMON="CSDL" SOTRINH="3" DIEM="6">

- Nếu muốn sử dụng giá trị số trình =>phải convert từ xâu sang số trong môi trường lập trình.

6 - Chú thích – comments

- Không được kiểm duyệt cú pháp và hiển thị

<!--lời chú giải-->

- Không được dùng -- hay - trong nội dung chú thích vì gây nhầm lẫn cho trình dịch XML
- Không đặt chú thích trong nội dung thẻ
  - VD sai <Sach <!--Thông tin sach-->>CSDL</Sach>
- Chú giải không được lồng nhau
- Chú giải có thể bỏ qua cả 1 đoạn

```xml
<?xml version="1.0" encoding="utf-8" ?>
<THUVIEN>
   <SACH>
     <MASACH>001</MASACH>
     <TENSACH>Cơ sở dữ liệu</TENSACH>
   </SACH>
  <!--Bỏ qua đoạn này
   <SACH>
     <MASACH>000</MASACH>
     <TENSACH>xml</TENSACH>
   </SACH>
   -->
</THUVIEN>
```

7 - Các chỉ thị xử lý

- Một chỉ thị xử lý là đoạn thông tin có ý nghĩa cho ứng dụng sử dụng tài liệu XML. Các lệnh được chuyển tới tài liệu nhờ bộ phân tích cú pháp.
- Ứng dụng có thể chuyển những lệnh này tới ứng dụng khác và tự thông dịch các lệnh
- Các lệnh xử lý tuân theo định dạng chung

<?xml-stylesheet type="text/xsl"?>

Tất cả các chỉ thị xử lý phải bắt đầu với dấu <? Và kết thúc ?>

8 - Phân loại dữ liệu ký tự giữa các thẻ

- Phần tài liệu giữa thẻ mở và thẻ đóng được định nghĩa là dữ liệu ký tự. Dữ liệu ký tự có thể là ký tự hợp lệ (unicode) bất kỳ ngoại trừ các ký hiệu nháy đơn, nháy kép, >,<. Thường được dùng các thực thể thay thế

Tên thực thể	Ký tự
<	<
>	>
&	&
"	"
'	'

9 - Các thực thể

- Các thực thể là các đơn vị lưu trữ của XML. Một thực thể có thể là 1 cụm từ hay được sử dụng, một ký tự bàn phím, 1 file, 1 bản ghi CSDL hay 1 mục dữ liệu.

- Các thực thể được sử dụng trong tài liệu để tránh phải gõ lại những đoạn văn bản dài lặp đi lặp lại trong tài liệu.
- **Các thực thể chung**
  - Các thực thể có thể xuất hiện mọi nơi trong tài liệu XML được gọi là các thực thể chung
  - Các thực thể có thể khai báo bên trong hay bên ngoài XML. Các thực thể bên trong chỉ tồn tại trong tài liệu mà nó được khai báo. Các thực thể bên ngoài được lưu trữ bên ngoài tài liệu và được tham chiếu
  - Cú pháp

<!ENTITY tenthucthe “Tài liệu được tham chiếu bởi thực thể”>

VD: <!ENTITY tenlop “KTPM-Khoa CNTT”>

Đây là thực thể nội

- Các thực thể ngoại sử dụng 1 định danh để chỉ ra 1 đơn vị lưu trữ bên ngoài tài liệu. Có 2 loại định danh thực thể bên ngoài
  - SYSTEM: nơi lưu trữ là máy cá nhân hay mạng cục bộ
  - PUBLIC: nới lưu trữ là mạng công cộng
- Ví dụ:

<!ENTITY tenlop SYSTEM “test.txt”>

  - Bộ xử lý XML tham chiếu thực thể thay thế bởi chuỗi trong file test.txt. File test.txt chứa phần tài liệu được thay thế khi thực thể được tham chiếu đến
  - Từ khóa SYSTEM chuyển hướng nơi tìm kiếm file
- Cú pháp lời gọi tham chiếu thực thể

&Tên_thực_thể;

VD - Giả sử 1 địa chỉ được lưu lại như 1 thực thể trong 1 file, mỗi lần viết 1 lá thư trong XML chúng ta sẽ gọi địa chỉ đó tên như trong ví dụ sau

```
<LATHU>
    &diachi;
    <NGUOINHAN>Trần Lan Anh</NGUOINHAN>
    <NOIDUNG>Xin chào bạn</NOIDUNG>
    <NGUOIGUI>Nguyễn Trung Hiếu</NGUOIGUI>
</LATHU>
```

- Các quy tắc tham chiếu thực thể
  - Các thực thể phải được khai báo trong 1 tài liệu XML trước khi chúng được tham chiếu
  - Không có dấu cách trong tham chiếu thực thể. VD: ‘& dia chi’
  - Đoạn tài liệu mà thực thể tham chiếu phải là 1 tài liệu XML hợp khuôn dạng

- Các thuộc tính của thể cũng có thể tham chiếu tới các thực thể
- VD <CLIENT="&APTECH;" PRODUCT="&ID;" QUANLITY="15">
- Chú ý – Phần tài liệu tham chiếu không nên chứa ký tự < vì nó có thể gây lỗi trong phần tử khi được thay thế

10 - Khai báo DOCTYPE

- Phần khai báo <!DOCTYPE […]> được đặt sau khai báo XML trong tài liệu XML
- Cú pháp:

```
<?xml version="1.0"?>
<!DOCTYPE myDoc[
    …Khai báo các thực thể…
]>
<myDoc>
    Nội dung file XML
</myDoc>
```

- **VD- cho bảng dữ liệu khách hàng –** Xây dựng file XML sử dụng thực thể

STT	Họ tên	Địa chỉ	Điện thoại
1	Trần Lan Anh	105 Tầng 1 – C2 – Mỹ đình 2	2342353454
2	Vũ Hoàng Minh	205 Tầng 1 – C2 – Mỹ đình 2	2342345235
3	Vũ Hoàng Linh	305 Tầng 1 – C2 – Mỹ đình 2	4563543545
4	Châu Huệ Mẫn	305 Tầng 1 – C2 – Mỹ đình 2	1212124344

```xml
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE DSKHACHHANG[
  <!ENTITY tang1 "105 Tầng 1 - C2 - Mỹ đình 2">
  <!ENTITY tang2 "205 Tầng 1 - C2 - Mỹ đình 2">
  <!ENTITY tang3 "305 Tầng 1 - C2 - Mỹ đình 2">
]>
<!--DSKHACHHANG LÀ NODE GỐC-->
<DSKHACHHANG>
  <!--Node KHACHHANG chứa thông tin chi tiết-->
  <KHACHHANG>
    <HOTEN>Trần Lan Anh</HOTEN>
    <DIACHI>&tang1;</DIACHI>
    <DIENTHOAI>2342353454</DIENTHOAI>
  </KHACHHANG>
  <KHACHHANG>
    <HOTEN>Vũ Hoàng Minh</HOTEN>
    <DIACHI>&tang2;</DIACHI>
    <DIENTHOAI>2342345235</DIENTHOAI>
  </KHACHHANG>
  <KHACHHANG>
    <HOTEN>Vũ Hoàng Linh</HOTEN>
    <DIACHI>&tang3;</DIACHI>
    <DIENTHOAI>4563543545</DIENTHOAI>
  </KHACHHANG>
  <KHACHHANG>
    <HOTEN>Châu Huệ Mẫn</HOTEN>
    <DIACHI>&tang3;</DIACHI>
    <DIENTHOAI>1212124344</DIENTHOAI>
  </KHACHHANG>
</DSKHACHHANG>
```

II. LÝ THUYẾT - XML Schema

– Lược đồ mô tả cấu trúc file XML

- Một lược đồ là 1 tập các luật để ràng buộc cấu trúc và truyền tải tập thông tin của các tài liệu XML. Một lược đồ mô tả một mô hình cho toàn bộ các tài liệu, mô tả cách đánh dấu dữ liệu và chỉ rõ sự sắp xếp có thể của các thẻ và văn bản trong 1 tài liệu hợp lệ.
- Một lược đồ có thể xem như 1 bộ từ vựng chung để trao đổi tài liệu giữa những tổ chức khác nhau.
- Ví dụ Các thông tin về sinh viên như hodem, ten, ngaysinh (từ ngày x đến ngày y), gioitinh (Nam hoặc nữ) thông qua ràng buộc dữ liệu ở lược đồ để kiểm tra cú pháp các thông tin này

```xml
<?xml version="1.0" encoding="utf-8" ?>
<DSSV>
  <sinhvien>
    <hodem>Phạm Tuấn</hodem>
    <ten>Đạt</ten>
    <Ngaysinh>6/12/2000</Ngaysinh>
    <gioitinh>Nam</gioitinh>
  </sinhvien>
</DSSV>
```

1 - Các kiểu dữ liệu

- xs:ID – kiểu định danh – thường sử dụng cho mã duy nhất
- xs:boolean – kiểu logic – true/false
- xs:binary – kiểu nhị phân
- xs:date – kiểu ngày tháng - ví dụ 2017 – 02- 19
- xs:number – kiểu số nguyên/thực
- xs:string – kiểu xâu ký tự
- xs:interger – số nguyên
- xs:decimal – số thực
- xs:real – số thực có mũ vd: 3.4E4
- xs:time – kiểu thời gian
- …

2 - Các kiểu dữ liệu do người dùng tự tạo

- Kiểu đơn giản – các phần tử kiểu đơn giản là các phần tử mà nội dung chỉ chứa dữ liệu dạng text. Dữ liệu dạng text là các dữ liệu được hỗ trợ bởi lược đồ như integer, real, decimal,…
- Chúng ta có thể tạo 1 kiểu dữ liệu mới với sự hỗ trợ của các kiểu dữ liệu xây dựng sẵn và các ràng buộc bổ sung nếu cần
- Cú pháp:

```
<xs:simpleType name="tên_kiểu">
   <xs:restriction base="tên_kiểu_cơ_sở">
      <!--Khai báo các ràng buộc dữ liệu-->
   </xs:restriction>
</xs:simpleType>
```

- Vd: Định nghĩa kiểu dữ liệu mới có tên diemType
  Kiểu dữ liệu cơ sở của diemType là integer với giá trị nhỏ nhất là 0 và lớn nhất là 10 của mỗi phần tử thuộc kiểu đó

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="XMLSchema1"
    targetNamespace="http://tempuri.org/XMLSchema1.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema1.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema1.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
  <xs:simpleType name="diemType">
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"></xs:minExclusive>
      <xs:maxExclusive value="10"></xs:maxExclusive>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

3 - Định nghĩa kiểu phần tử

```
<xs:element name="tenphantu" type="tenkieu"></xs:element>
```

- Ví dụ sử dụng kiểu phàn tử

  < xs:element name=”hoten” type=” xs:string”>

  < xs:element name=”ngaysinh” type=” xs:date”>

  < xs:element name=”HSLuong” type=” xs:double”>

- Ví dụ Sử dụng kiểu dữ liệu định nghĩa kiểu phần tử

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="XMLSchema1"
    targetNamespace="http://tempuri.org/XMLSchema1.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema1.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema1.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
  <xs:simpleType name="diemType">
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="0"></xs:minExclusive>
      <xs:maxExclusive value="10"></xs:maxExclusive>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="diemtoan" type="diemType"></xs:element>
  <xs:element name="diemly" type="diemType"></xs:element>
  <xs:element name="diemhoa" type="diemType"></xs:element>
</xs:schema>
```

- Các thuộc tính của phần tử element
  - default – giá trị ngầm định cho phần tử
  - fixed – giá trị cố định của phần tử

- name – tên phần tử
- type – kiểu dữ liệu của phần tử
- id – định danh của phần tử
- maxOccurs – số lần xuất hiện tối đa
- minOccurs – số lần xuất hiện tối thiểu
- ref – tham chiếu tới phần tử khác

4 - Các ràng buộc trong lược đồ

- Ràng buộc về kiểu dữ liệu cho phép kiểm soát phạm vi và định dạng của dữ liệu
- Để ràng buộc về kiểu dữ liệu cho phần tử ta sử dụng phần tử restriction. Phần tử này có các thuộc tính:
  - base – xác định kiểu cơ sở cho phần tử
  - id – định danh của kiểu phần tử
- Cú pháp:

  <xs:restriction base="kiểu_cơ_sở">

  <!--Các ràng buộc-->

  <xs:restriction>
- Các ràng buộc bao gồm:
  - enumration – danh sách các giá trị hợp lệ
  - fractionDigits – số chữ số phần thập phân, >=0
  - length – số lượng ký tự, >=0
  - maxExclusive – Cận trên của giá trị kiểu số (<=)
  - minExclusive – Cận dưới của giá trị kiểu số (>=)
  - maxLength – số kỹ tự tối đa của 1 chuỗi (>=0)
  - minLength – số ký tự tối thiểu của 1 chuỗi (>=0)
  - maxInclusive – Cận trên của giá trị kiểu số (<)
  - minInclusive – Cận dưới của giá trị kiểu số (>)
  - pattern – Chính xác Các ký tự hợp lệ
  - totalDigits – chính xác số các chữ số được chấp nhận (>=0)
  - whiteSpace – định nghĩa ràng buộc dấu cách
- **Ràng buộc kiểu giới hạn**

  <xs:minExclusive value="giá trị min">

  <xs:maxExclusive value="giá trị max">
- **Ràng buộc kiểu liệt kê**

  <xs:enumration value="giá trị 1">

  <xs:enumration value="giá trị 2">

  VD: giới tính chỉ là Nam hoặc Nữ

```
<xs:simpleType name=”gtType”>
        <xs:restriction base=”xs:string”>
                <xs:enumration value=”Nam”>
                <xs:enumration value=”Nữ”>
        </xs:restriction>
</xs:simpleType>
```

- **Ràng buộc kiểu so mẫu**

<xs:pattern value=”mẫu dữ liệu”>

Ký hiệu	Mô tả
[0-9]	một ký tự số từ 0-9
[a..z]	một ký tự từ a đến z
|	chọn mẫu này hoặc mẫu khác
\w	ký tự thay thế phải là 1 chữ cái
\d	ký tự thay thế phải là 1 chữ số
?	quy định số lần xuất hiện 0 hoặc 1 lần
*	quy định số lần xuất hiện 0 hoặc nhiều lần
{n}	quy định số lần xuất hiện chính xác n lần

VD: tạo kiểu password chỉ chứa 4 ký tự: đầu tiên là 1 ký tự a hoặc b, tiếp theo là 3 ký tự số

<xs:pattern value=”[ab]\d{3}”>

5 - Bảng từ vựng lược đồ xml

- Schema là phần tử gốc của tất cả các tài liệu lược đồ xml
- Phần tử schema có các phần tử con
  - annotation
  - attribute
  - attributeGroup
  - complexType
  - simpleType
  - element
  - group
  - notation
  - include
  - import

6 - Kiểu phức hợp – complexType

- Các phần tử có kiểu phức hợp là các phần tử mà nội dung của nó có thể chứa

- Các phần tử khác
- Thuộc tính
- Nội dung hỗn hợp (vừa có dữ liệu text, vừa chứa phần tử con)
- Là phần tử rỗng

- Cú pháp:

```
<xs:complexType name="tenkieu">
    <xs:sequence>
        <!--Định nghĩa nội dung phần tử-->
    </xs:sequence>
    <!--Định nghĩa các kiểu thuộc tính nếu có-->
<xs:complexType>
```

- **Phần tử attribute** – định nghĩa thuộc tính cho 1 phần tử. Nó có các thuộc tính con
  - default – giá trị ngầm định cho thuộc tính
  - fixed – giá trị cố định cho thuộc tính
  - name – tên thuộc tính
  - type – kiểu dữ liệu của thuộc tính
  - id – định danh thuộc tính
  - use – optional (không bắt buộc)/required (bắt buộc phải xuất hiện)
- Cú pháp

```
<xs:attribute name="tên" type="kiểu" use="required/optional">
        <!--Nội dung cần định nghĩa-->
</xs:attribute>
```

- VD định nghĩa thuộc tính doituong và ngay cho kiểu phần tử nhomtype

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="nhomType"
    targetNamespace="http://tempuri.org/nhomType.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/nhomType.xsd"
    xmlns:mstns="http://tempuri.org/nhomType.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
  <!--Mỗi nhomType là 1 kiểu phức hợp có các attribute chung cho mỗi nhóm, các element lặp lại-->
  <xs:complexType name="nhomType">
    <xs:sequence>
      <xs:element name="thoigian" type="xs:time" minOccurs="1" maxOccurs="1"></xs:element>
      <xs:element name="hạng" type="xs:double" minOccurs="1" maxOccurs="1"></xs:element>
      <xs:element name="diadiem" type="xs:string" minOccurs="1" maxOccurs="1"></xs:element>
      <xs:element name="baocao" type="xs:string" minOccurs="1" maxOccurs="1"></xs:element>
    </xs:sequence>
    <xs:attribute name="doituong" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="ngay" type="xs:date" use="required"></xs:attribute>
  </xs:complexType>
</xs:schema>
```

7 - Chỉ định giá trị mặc định cho phần tử hoặc thuộc tính

- fixed – gán giá trị bắt buộc cho phần tử
- default – gắn giá trị mặc định cho phần tử
- VD: với khai báo
  <xs:element name="abc" fixed="100$">
  => phần tử abc có giá trị bắt buộc 100$, Nếu <abc>120$</abc> sẽ bị báo lỗi
  <xs:element name="abc" default="100$">
  => phần tử abc có giá trị ngầm định 100$, Nếu <abc>120$</abc> thì abc sẽ có giá trị 120$, ngược lại <abc></abc> thì abc mặc định 100$

Tài liệu tham khảo:

- Sách, giáo trình chính:

[1]. Giáo trình dịch vụ web và ứng dụng/ Đại học KHTN – Đại học Quốc gia TP HCM, 2016.

[2]. Anura Guruge. Web service: Theory and Practice. Elsevier press 20017.

[3]. Scott Klein. Professional API- RESTfull Programming. Wiley Publishing, 2016.

- Sách, tài liệu tham khảo:

[1]. Alex Ferra, Mathew MacDonald. Programming .NET web services. O'Reilly

Media, 2012.

[2]. Mark D. Hansen. SOA Using .NET web services. Prentice Hall, 2011.