

Web ngữ nghĩa

Soạn bởi: Nguyễn Bá Ngọc

Chương 2

Hà Nội-2021

Chương 2.

XML & JSON

Nội dung

2.1. Căn bản XML

2.2. Xử lý dữ liệu XML

2.3. Căn bản JSON

2.4. Xử lý dữ liệu JSON

Nội dung

2.1. Căn bản XML

2.2. Xử lý dữ liệu XML

2.3. Căn bản JSON

2.4. Xử lý dữ liệu JSON

Vai trò của XML trong Web ngữ nghĩa

- Nhiều ngôn ngữ, công nghệ và công cụ được phát triển để hiện thực hóa ý tưởng Web ngữ nghĩa (WNN).
 - Ví dụ, Để biên soạn các ontologies, chia sẻ dữ liệu, v.v..
- XML có thể được sử dụng như định dạng lưu trữ
 - Phù hợp để chia sẻ giữa các hệ thống ứng dụng và trong môi trường Web.
- Định dạng XML được sử dụng phổ biến cho dữ liệu WNN.

XML và HTML

- Cùng được phát triển từ SGML
 - SGML - Standard Generalized Markup Language
 - HTML- là 1 ngôn ngữ đánh dấu, do TBL sáng tạo, ~1990
 - (Ngôn ngữ đánh dấu = *markup language*)
 - XML được thiết kế như 1 siêu ngôn ngữ - cho phép xây dựng các ngôn ngữ đánh dấu
 - Khả mở, có thể tự định nghĩa các thẻ mới
 - Bản đề xuất được gửi W3C năm 1996
- XML (đã) là 1 quy chuẩn W3C
 - Được cập nhật nhiều lần, phiên bản mới nhất là XML 1.1.

XML và HTML₍₂₎

HTML

*Thuận tiện cho người
đọc*

```
<h2>Foundations of Semantic Web  
Technologies</h2>  
<i>by <b>Pascal Hitzler</b>, <b>  
Markus Krötzsch</b> and <b>  
Sebastian Rudolph</b><br>  
CRC Press 2009<br>  
ISBN 978-1-4200-9050-5<br>
```

XML

*Thuận tiện cho xử lý
bằng máy tính*

```
<book>  
  <title>Foundations of Semantic  
  Web Technologies</title>  
  <author>Pascal Hitzler</author>  
  <author>Markus Krötzsch</author>  
  <author>Sebastian Rudolph</author>  
  <publisher>CRC Press</publisher>  
  <year>2009</year>  
  <ISBN>978-1-4200-9050-5</ISBN>  
</book>
```

Các điểm tương đồng:

- Cùng sử dụng các thẻ
- Các thẻ được phép lồng nhau

HTML vs XML: Các thành phần cấu trúc

- Ví dụ, tình huống trích xuất thông tin tác giả:
 - HTML: Không chứa cấu trúc thông tin => Khó thực hiện
 - XML: Có chứa các cấu trúc thông tin => Dễ thực hiện hơn
 - Ví dụ, có thể trích xuất thông tin tác giả theo thẻ <author> xuất hiện bên trong thẻ <book>
- Suy diễn tự động:
 - Thông tin trong thẻ author nằm trong thẻ book là thông tin về tác giả của sách
 - => không sử dụng đến các đại lượng thống kê, hoặc các phương pháp học máy

HTML vs XML: Định dạng

- HTML từ ban đầu được thiết kế cho mục đích hiển thị
 - Chủ yếu mô tả cách hiển thị tài liệu.
- XML: Tách biệt nội dung và cách hiển thị
 - Cùng 1 nội dung có thể được hiển thị theo nhiều cách
 - Cách hiển thị được mô tả trong các tài liệu sử dụng các quy chuẩn khác (ví dụ, CSS, XSL).

HTML vs XML: Thẻ

- Tất cả các tài liệu HTML đều sử dụng chung 1 tập thẻ (HTML)
 - Tập thẻ HTML là hữu hạn và có ý nghĩa được thiết lập bằng quy chuẩn
 - Quy định các thuộc tính hiển thị: phông chữ, màu sắc, v.v.
- Các tài liệu XML khác nhau có thể sử dụng các tập thẻ hoàn toàn khác nhau
 - Tập thẻ XML không giới hạn: Người dùng có tự định nghĩa các thẻ mới.
 - XML là 1 siêu ngôn ngữ đánh dấu - Cho phép tạo các ngôn ngữ đánh dấu, định nghĩa các thẻ mới.

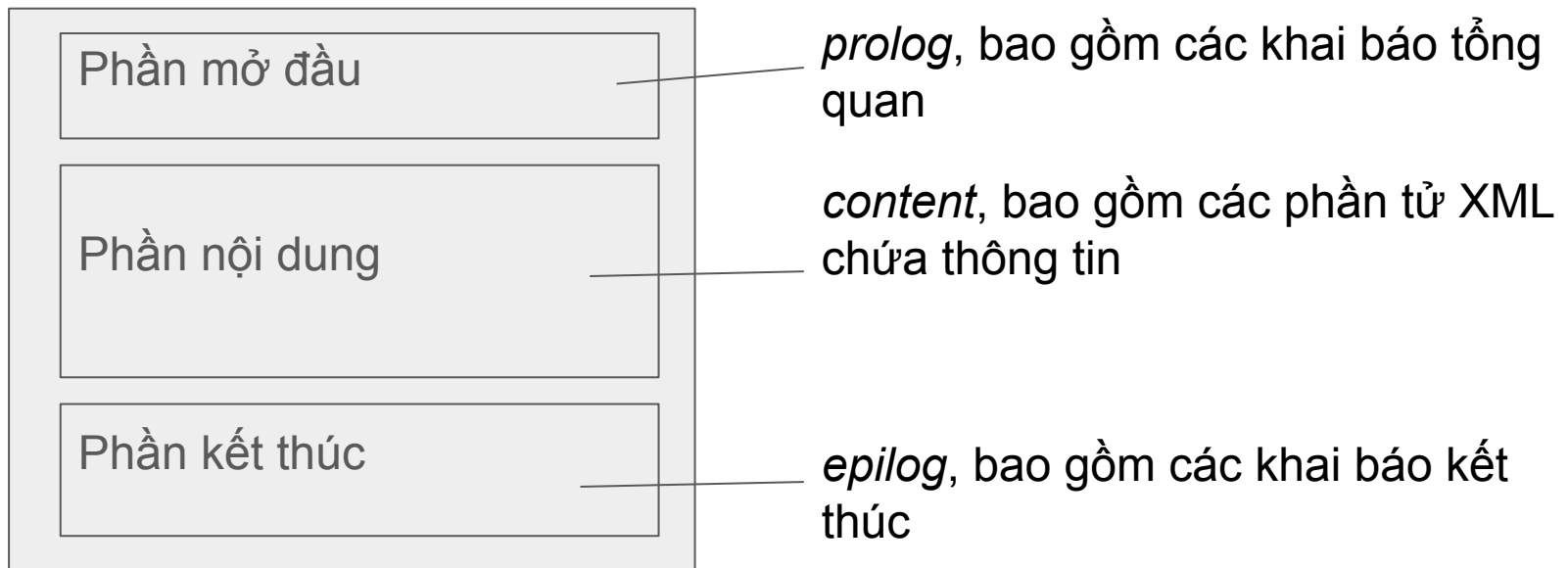
Các bộ từ vựng XML

- Các ứng dụng phần mềm khác nhau phải thống nhất trên các bộ từ vựng dùng chung để giao tiếp và tương tác
- Các bộ từ vựng XML tiêu biểu:
 - MathML (Toán học)
 - BSML (Tin Sinh)
 - HRML (Tài nguyên nhân lực)
 - RSS (Tin tức)
 - SVG (Đồ họa vec-tơ)
 - ...

Cấu trúc tài liệu XML

Cấu trúc khái quát

Một tài liệu XML thường có cấu trúc gồm 3 phần:



Tài liệu XML còn được biểu diễn bằng cấu trúc dạng cây

Tài liệu XML: Phần mở đầu

- Phần mở đầu bao gồm
 - Một khai báo XML
 - (Có thể có) các liên kết tới các tài liệu khác mô tả cấu trúc



```
<?xml version="1.0" encoding="UTF-16"?>
```

```
<!DOCTYPE book SYSTEM "book.dtd">
```

Tài liệu XML: Phần nội dung

- Phần nội dung bao gồm nhiều phần tử lưu thông tin
 - Ví dụ, sách, tác giả, nhà xuất bản, v.v..
- Mỗi phần tử bao gồm

- Một thẻ mở
- Nội dung thông tin
- Một thẻ đóng



```
graph TD; A[Một thẻ mở] --> B["<title>"]; C[Nội dung thông tin] --> D["The foundations of Semantic Web Technologies"]; E[Một thẻ đóng] --> F["</title>"]
```

`<title>The foundations of Semantic Web Technologies</title>`

Các phần tử XML

- Tên thẻ chỉ cần tuân theo quy tắc đặt tên
 - Ký tự đầu tiên phải là 1 chữ cái, dấu gạch dưới (_), hoặc chấm phẩy (;).
- Nội dung là tất cả những gì nằm giữa cặp thẻ đóng và thẻ mở
 - Có thể là văn bản, các phần tử khác, hoặc không có gì
- Phần tử không có nội dung được gọi là phần tử rỗng và có thể được viết tắt:
 - `<element/>` = `<element></element>`

Các thuộc tính XML

- Thuộc tính là cặp tên-giá trị nằm trong thẻ (mở) của 1 phần tử

```
<order id="123"
      cust="Nguyen Van A">
  <item id="a123" quant="10" />
  <item id="b567" quant="20" />
</order>
```

- Các phần tử rỗng vẫn có thể có thuộc tính
- Các thuộc tính có thể được thay thế bằng các thẻ

```
<order>
  <id>123</id>
  <cust>Nguyen Van A</cust>
  <item>
    <id>a123</id>
    <quant>10</quant>
  </item>
  <item>
    <id>b567</id>
    <quant>20</quant>
  </item>
</order>
```

Lựa chọn sử dụng thuộc tính hay phần tử con?

Một số thành phần khác trong tài liệu XML

- Các chú thích
 - Cho người đọc, máy tính bỏ qua khi xử lý
 - `<!-- Đây là một chú thích -->`
- Các lệnh / điều khiển xử lý (PIs)
 - Được sử dụng để cung cấp thông tin cho tiến trình xử lý
 - `<?stylesheet type="text/css" href="mystyle.css" ?>`

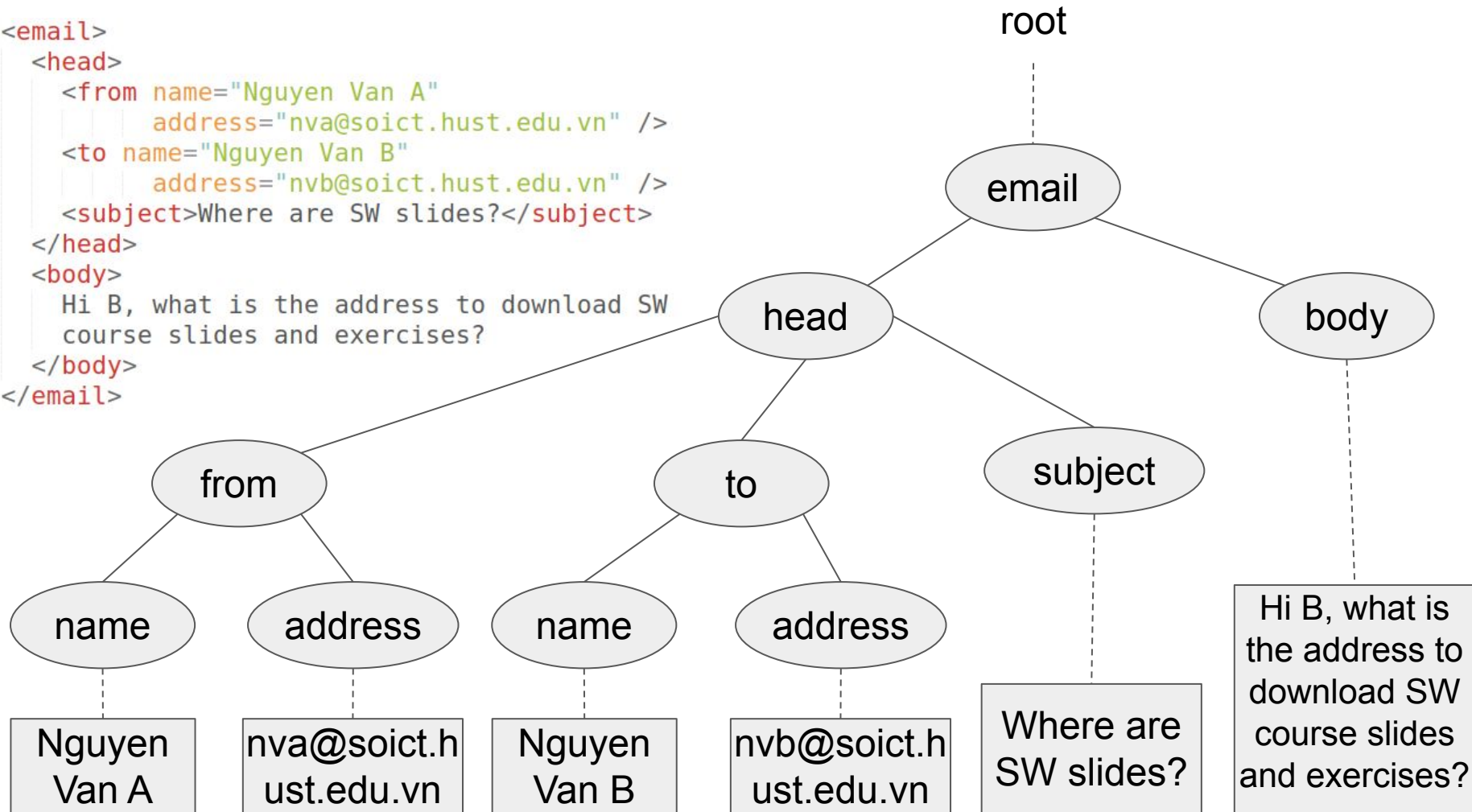
Mô hình cây của tài liệu XML

- Tài liệu XML có thể được biểu diễn như cây có nhãn và có thứ tự:
 - Có đúng 1 gốc
 - Không có chu trình
 - Mỗi nút không phải gốc có đúng 1 nút cha
 - Mỗi nút có 1 nhãn
 - Trật tự của các nút có ý nghĩa quan trọng
 - Tương ứng với trật tự phần tử, thay đổi trật tự => thay đổi ý nghĩa

Khác với các phần tử, thứ tự của các thuộc tính không quan trọng.

Ví dụ 2.1. Cây XML

```
<email>
  <head>
    <from name="Nguyen Van A"
          address="nva@soict.hust.edu.vn" />
    <to name="Nguyen Van B"
        address="nvb@soict.hust.edu.vn" />
    <subject>Where are SW slides?</subject>
  </head>
  <body>
    Hi B, what is the address to download SW
    course slides and exercises?
  </body>
</email>
```



Các thành phần cấu trúc trong tài liệu XML

Các thành phần cấu trúc

- Các quy tắc có thể được thiết lập cho tài liệu XML:
 - Giới hạn tên phần tử và tên thuộc tính
 - Giới hạn cấu trúc
 - Miền giá trị của thuộc tính
 - Các phần tử có thể hoặc phải xuất hiện bên trong các phần tử khác
- Với các ràng buộc được thiết lập, các tài liệu XML có thể là hợp lệ hoặc không hợp lệ
 - Tính hợp lệ của tài liệu XML được kiểm tra

Các thành phần cấu trúc₍₂₎

- Một tài liệu XML là hợp lệ nếu như
 - Đúng định dạng XML
 - Đúng cấu trúc theo quy ước
- Các ngôn ngữ mô tả cấu trúc tài liệu XML:
 - DTDs (Document Type Definition) cách tiếp cận đầu tiên, dựa trên SGML
 - XSD (XML Schema Definition) cách tiếp cận mới hơn, có khả năng diễn đạt mạnh hơn
 - Các phương án thay thế: RELAX NG và DSDs

Ví dụ 2.2. XML & DTD

```
<order orderId="o123"
      cust="Nguyen Van A">
  <item itemId="i123" quant="10" />
  <item itemId="i567" quant="20" />
</order>
```

Hợp lệ

```
<order orderId="o123"
      cust="Nguyen Van A">
  <item itemId="i123" quant="10" />
  <item itemId="i567" quant="20" />
  <item itemId="i222" />
</order>
```

Không hợp lệ

```
<!DOCTYPE order [
  <!ELEMENT order (item+)>
  <!-- ATTLIST order -->
  <!-- order attributes -->
  orderId ID #REQUIRED
  cust CDATA #REQUIRED
] >

<!-- ELEMENT item EMPTY -->
<!-- ATTLIST item -->
  itemId ID #REQUIRED
  quant CDATA #REQUIRED
]>
```

Chúng ta đã thiết lập các ràng buộc:

- Phần tử order phải chứa ít nhất 1 phần tử item và bắt buộc phải có 2 thuộc tính orderId và cust.
- item phải là phần tử rỗng và có chứa 2 thuộc tính: itemId và quant.

XSD - Lược đồ XML

- Lược đồ XML có khả năng diễn đạt cao hơn đáng kể so với DTD.
- Cú pháp dựa trên XML vì vậy có thể sử dụng các công cụ được phát triển cho XML để xử lý XSD.
- Khả năng tái sử dụng hoặc chi tiết hóa các lược đồ
- Hệ thống kiểu dữ liệu phong phú hơn đáng kể so với DTD
- Quy chuẩn Web cho lược đồ XML
 - W3C thông qua phiên bản đầu tiên năm 2001,
 - phiên bản mới nhất là 1.1, được thông qua năm 2012.

Ví dụ 2.3. XML & XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="order">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="item" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="itemId" type="xs:ID" use="required" />
            <xs:attribute name="quant" type="xs:int" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="orderId" type="xs:ID" use="required" />
      <xs:attribute name="cust" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

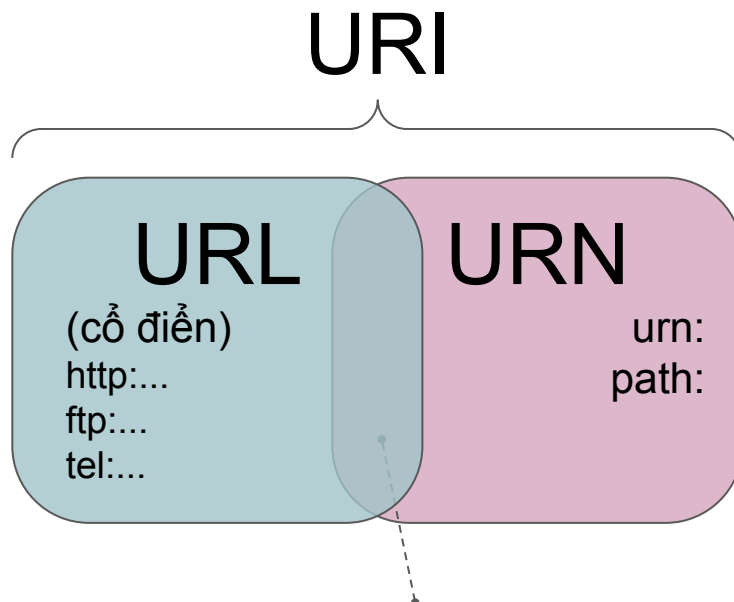
Không gian tên XML

Ý nghĩa của không gian tên

- Một cơ chế hiệu quả để tránh vấn đề trùng lặp tên
 - Các bộ từ vựng được phát triển độc lập dẫn đến rủi ro trùng lặp tên
 - Một tài liệu XML có thể sử dụng >1 bộ từ vựng
 - => sử dụng tiền tố kết hợp với tên để tránh trùng lặp
- Tiền tố gắn liền với URI
 - Được sử dụng trong DTD hoặc XSD
 - Ngữ cảnh của tên

URLs, URNs, URIs, ...

Các loại chuỗi định danh theo W3C



*URL cố định và duy nhất,
được sử dụng như tên*

[<https://www.w3.org/Addressing/URL/Addressing.html>]

Khai báo không gian tên

- Không gian tên được khai báo bên trong các phần tử
- Theo định dạng:
 - `xmlns:prefix="location"`
 - Trong đó location là URI của DTD hoặc XSD.
 - prefix là tiền tố (ví dụ m, svg, xlink, v.v.), là quy ước viết ngắn gọn cho location.
 - Nếu để trống prefix thì location được hiểu là không gian tên mặc định (cho những tên không có tiền tố).

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:m="http://www.w3.org/1998/Math/MathML"
      xmlns:svg="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
</html>
```

Mặc định, không có tiền tố

Đặt tên ngắn cho các chuỗi định danh:

m, svg, xlink

Nội dung

2.1. Căn bản XML

2.2. Xử lý dữ liệu XML

2.3. Căn bản JSON

2.4. Xử lý dữ liệu JSON



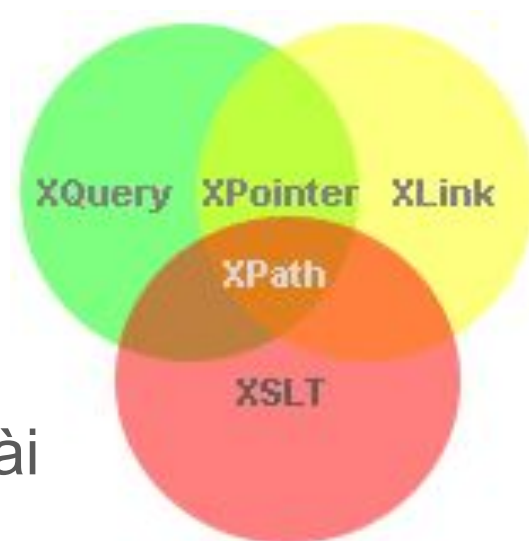
Truy xuất dữ liệu XML

Ngôn ngữ truy vấn XML

- Ngôn ngữ để thao tác dữ liệu, tương tự như lấy dữ liệu từ CSDL quan hệ bằng SQL
 - Đối với XML: XQuery, XQL, XML-QL
- Thành phần cốt lõi của các ngôn ngữ truy vấn dữ liệu XML là các biểu thức đường dẫn (XPath)
 - Xác định một phần tử/nút hoặc một tập phần tử/nút trong tài liệu XML
- Hữu ích để trích xuất dữ liệu từ tài liệu XML

XPath

- Thành phần cốt lõi của nhiều công nghệ XML
- Cho phép lựa chọn các nút trong các tài liệu XML
 - Theo mô hình dữ liệu dạng cây của XML
- Các phiên bản
 - Xpath 1.0 (1999) được hỗ trợ rộng rãi
 - Xpath 2.0 (2007)
 - Xpath 3.1 (2017)



[w3schools]

Khả năng diễn đạt mạnh hơn với các cấu trúc mới

Ví dụ 2.4. XML & XPath

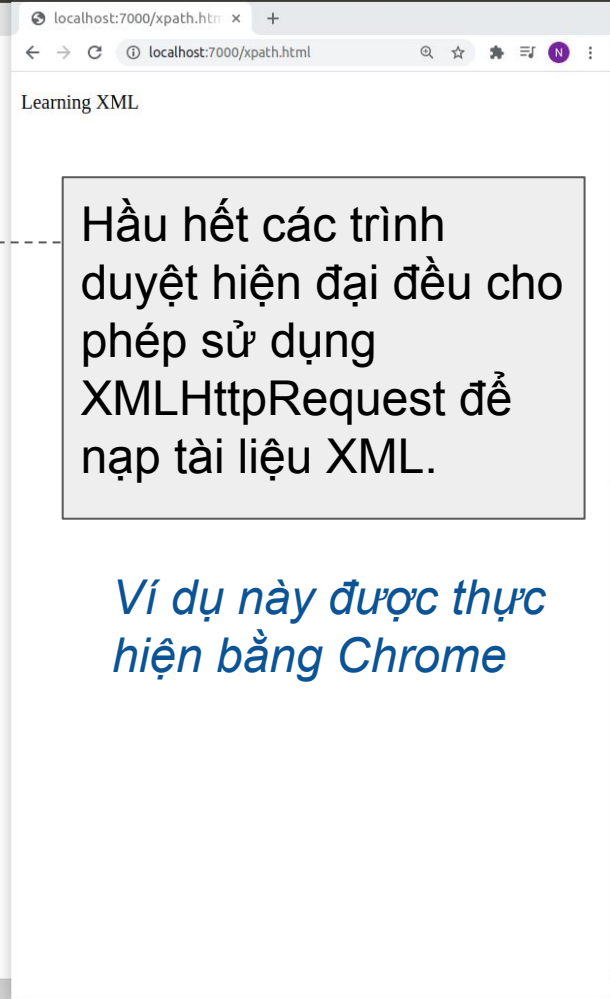
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<bookstore>
  <book category="programming">
    <title lang="en">The Go Programming Language</title>
    <author>Alan A.A. Donovan</author>
    <author>Brian W. Kernighan</author>
    <year>2016</year>
    <price>30.00</price>
  </book>
  <book category="semweb">
    <title lang="en">Foundations of Semantic Web Technologies</title>
    <author>Pascal Hitzler</author>
    <author>Markus Krötzsch</author>
    <author>Sebastian Rudolph</author>
    <year>2009</year>
    <price>59.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

/bookstore/book[1] - Lấy phần tử *book* đầu tiên là con của *bookstore*
/bookstore/book[last()] - Lấy *book* cuối cùng là con của *bookstore*
//title[@lang='en'] - Lấy các phần tử *title* có thuộc tính *lang* có giá trị là *en*
/bookstore/book[price > 35] - Lấy *book* là con của *bookstore* và có *price* với giá trị lớn hơn 35.

Ví dụ 2.5. XML & XPath trong máy khách

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="demo"></p>
5 <script>
6 var xhttp = new XMLHttpRequest(); -----
7 xhttp.onreadystatechange = function() {
8     if (this.readyState == 4 && this.status == 200) {
9         showResult(xhttp.responseXML);
10    }
11 };
12 xhttp.open("GET", "bookstore.xml", true);
13 xhttp.send();
14 function showResult(xml) {
15     var txt = "";
16     path = "/bookstore/book[last()]/title"
17     var nodes = xml.evaluate(path,
18         xml, null, XPathResult.ANY_TYPE, null);
19     var result = nodes.iterateNext();
20     while (result) {
21         txt += result.childNodes[0].nodeValue + "<br>";
22         result = nodes.iterateNext();
23     }
24     document.getElementById("demo").innerHTML = txt;
25 }
26 </script>
```



Biến đổi tài liệu XML: XSLT

Ví dụ 2.6. Hiển thị các tài liệu XML

```
<order orderId="o123"
      cust="Nguyen Van A">
  <item itemId="i123" quant="10" />
  <item itemId="i567" quant="20" />
</order>
```

Một tài liệu XML có thể được hiển thị theo nhiều cách

Order: o123

Customer: Nguyen Van A

ItemId	Quantity
i123	10
i567	20

Order: o123 Nguyen Van A

i123 x 10
i567 x 20

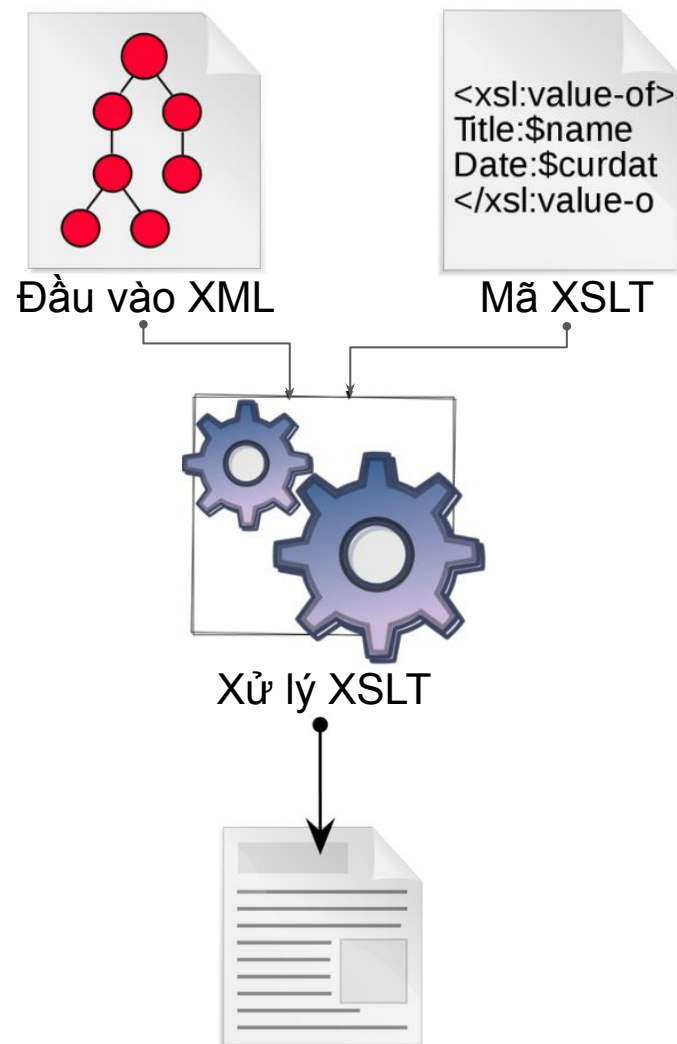
Ý tưởng: Sử dụng tầng phong cách hiển thị để biến đổi một cây XML thành 1 tài liệu HTML hoặc cây XML khác.

Tầng phong cách hiển thị

- Có thể được mô tả bằng nhiều ngôn ngữ khác nhau
 - Ví dụ, CSS2 (Cascading Style Sheets mức 2)
 - XSL (Extensible Stylesheet Language)
- XSL bao gồm
 - Một ngôn ngữ biến đổi (XSLT)
 - XSLT 1.0, quy chuẩn, 1999
 - XSLT 2.0, 2009
 - XSLT 3.0, 2017 => Mới nhất
 - Một ngôn ngữ định dạng
 - Cả 2 đều là các công nghệ XML

Biến đổi XSL (XSLT)

- XSLT thiết lập các quy tắc để biến đổi một tài liệu XML thành
 - 1 tài liệu XML khác
 - Tài liệu HTML
 - văn bản
- Đầu ra có thể sử dụng cùng DTD hoặc lược đồ, hoặc 1 bộ từ vựng hoàn toàn khác
- XSLT có thể được sử dụng độc lập với ngôn ngữ định dạng



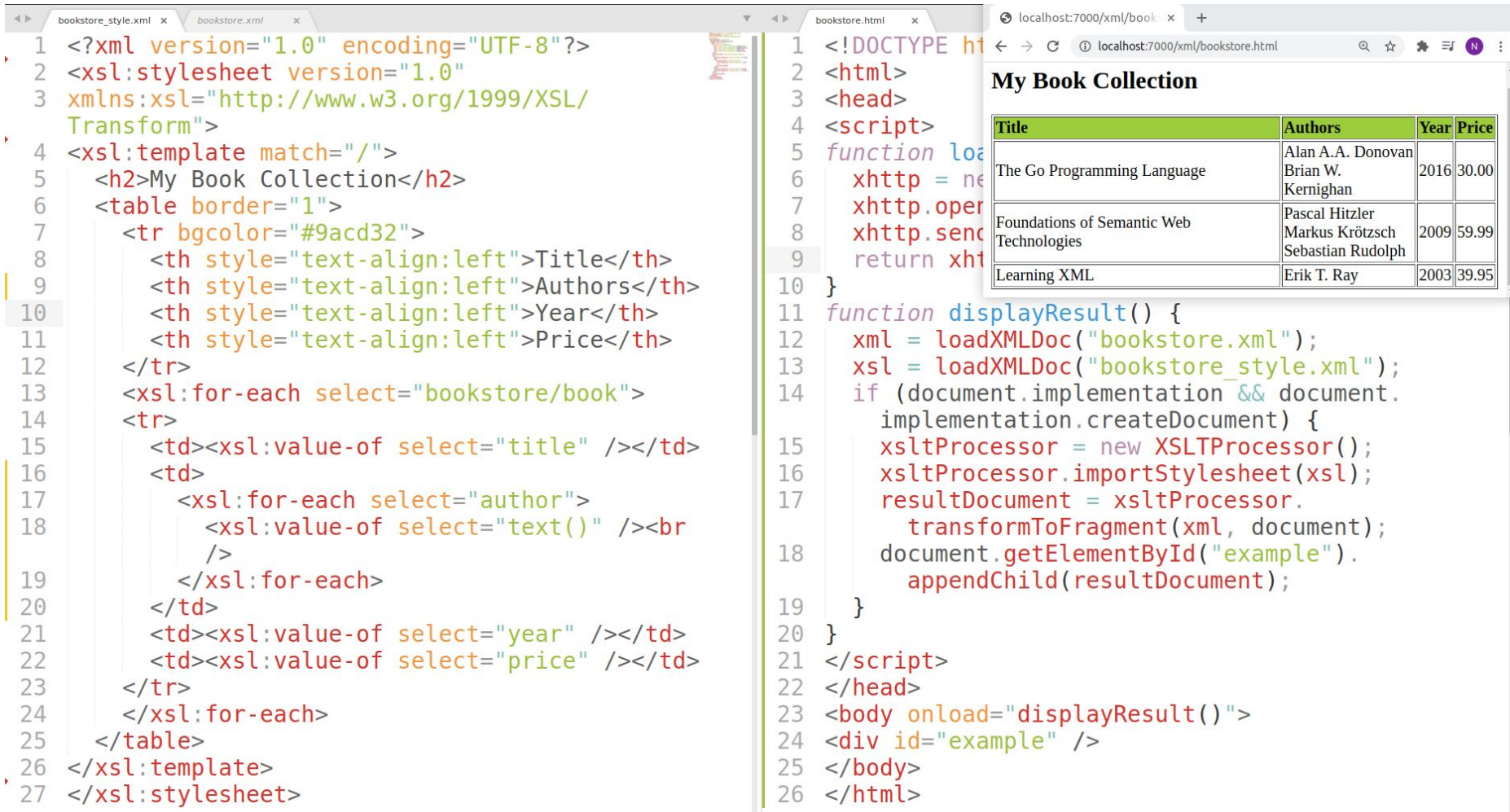
Một số trường hợp sử dụng XSLT

- Chuyển dữ liệu & siêu dữ liệu từ một biểu diễn XML sang biểu diễn khác
- Chia sẻ thông tin giữa các ứng dụng sử dụng các lược đồ khác nhau
- Xử lý các nội dung XML để nhúng vào một chương trình hoặc CSDL
- Hiển thị XML như HTML

```
<?xml version="1.0"
<xsl:stylesheet xmlns
<!-- created 2005-12-12-->
<xsl:include href="xslt
<xsl:output method="xml"
<xsl:template match="/">
<root>
  Heuristic:<xsl:value-of
    <p>The leading manufact
  </root>
</xsl:template>
</xsl:stylesheet>
```

XSLT

Ví dụ 2.7. Biến đổi XML trong máy khách



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0"
3 xmlns:xsl="http://www.w3.org/1999/XSL/
  Transform">
4 <xsl:template match="/">
5   <h2>My Book Collection</h2>
6   <table border="1">
7     <tr bgcolor="#9acd32">
8       <th style="text-align:left">Title</th>
9       <th style="text-align:left">Authors</th>
10      <th style="text-align:left">Year</th>
11      <th style="text-align:left">Price</th>
12    </tr>
13    <xsl:for-each select="bookstore/book">
14      <tr>
15        <td><xsl:value-of select="title" /></td>
16        <td>
17          <xsl:for-each select="author">
18            <xsl:value-of select="text()" /><br />
19          </xsl:for-each>
20        </td>
21        <td><xsl:value-of select="year" /></td>
22        <td><xsl:value-of select="price" /></td>
23      </tr>
24    </xsl:for-each>
25  </table>
26 </xsl:template>
27 </xsl:stylesheet>
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5   function loadXMLDoc(xml) {
6     xmlhttp = new XMLHttpRequest();
7     xmlhttp.open("GET", xml, true);
8     xmlhttp.send();
9     return xmlhttp.responseText;
10  }
11  function displayResult() {
12    xml = loadXMLDoc("bookstore.xml");
13    xsl = loadXMLDoc("bookstore_style.xml");
14    if (document.implementation && document.
15      implementation.createDocument) {
16      xsltProcessor = new XSLTProcessor();
17      xsltProcessor.importStylesheet(xsl);
18      resultDocument = xsltProcessor.
19        transformToFragment(xml, document);
20      document.getElementById("example").
21        appendChild(resultDocument);
22    }
23  }
24 </script>
25 </head>
26 <body onload="displayResult()">
27   <div id="example" />
28 </body>
29 </html>
```

Title	Authors	Year	Price
The Go Programming Language	Alan A.A. Donovan Brian W. Kernighan	2016	30.00
Foundations of Semantic Web Technologies	Pascal Hitzler Markus Krötzsch Sebastian Rudolph	2009	59.99
Learning XML	Erik T. Ray	2003	39.95

Một số đặc điểm của XSLT

- Tài liệu XSLT là tài liệu XML
 - XSLT được xây dựng trên cơ sở XML
- Tài liệu XSLT xác định 1 khuôn mẫu
 - Trong ví dụ 2.7 là 1 mẫu tài liệu HTML với những điểm đánh dấu để chèn nội dung vào
- **xsl:value-of** đọc nội dung của nút XML và đưa vào tài liệu đầu ra
 - Đưa nội dung vào khuôn mẫu

Tổng kết XML

- XML là 1 siêu ngôn ngữ đánh dấu cho phép người dùng tự định nghĩa các thẻ
- XML tách biệt cấu trúc và nội dung với định dạng hiển thị
- XML là 1 công nghệ tiêu chuẩn để biểu diễn và trao đổi thông tin có cấu trúc trên Web
- Các ngôn ngữ truy vấn XML
- Các ngôn ngữ biến đổi XML

Nội dung

2.1. Căn bản XML

2.2. Xử lý dữ liệu XML

2.3. Căn bản JSON

2.4. Xử lý dữ liệu JSON



JSON lựa chọn thay thế cho XML

- JSON là 1 lựa chọn khác nhẹ hơn XML để trao đổi dữ liệu
 - Cú pháp đơn giản, không chứa ngữ nghĩa
 - ... và không có lược đồ, không có ngôn ngữ biến đổi
- JSON = JavaScript Object Notation
 - Hầu hết các ngôn ngữ lập trình đều có thể đọc dữ liệu JSON và tạo các cấu trúc dữ liệu
- Được định nghĩa trong RFC 4627, IETF, 2006
 - Phiên bản hiện tại là RFC 8259, 2017
- Xem thông tin chi tiết hơn ở <http://json.org>

Đơn giản là tốt

Ví dụ 2.8. Tài liệu JSON

1 đối tượng JSON với 5 cặp khóa-giá trị

```
{"firstName": "John",  
  "lastName" : "Smith",  
  "age"       : 25,  
  "address"   :  
    {"streetAdr" : "21 2nd Street",  
     "city"      : "New York",  
     "state"     : "NY",  
     "zip"       : "10021"},  
  "phoneNumber":  
    [ {"type" : "home",  
       "number": "212-555-1234"},  
      {"type" : "fax",  
       "number" : "646-555-4567"} ]
```

- Khóa là chuỗi ký tự
- Giá trị là số, chuỗi ký tự, đối tượng hoặc mảng
- Các đối tượng được đóng trong các ngoặc nhọn
- Các mảng/danh sách được đóng trong các ngoặc vuông

JSON-LD

- Một quy chuẩn của W3C để biểu diễn dữ liệu RDF như các đối tượng JSON
- Các phiên bản
 - JSON-LD 1.0, 2014
 - JSON-LD 1.1, 2020

Tham khảo:

[1] <http://markbirbeck.com/2009/04/20/rdfj-semantic-objects-in-json/>

[2] <https://www.w3.org/TR/2012/WD-json-ld-syntax-20120712/>

[3] <https://www.w3.org/TR/2014/REC-json-ld-20140116/>

[4] <https://www.w3.org/TR/json-ld11/>

Tổng quan về JSON-LD

- Đưa ngữ nghĩa vào JSON thông qua ngữ cảnh (context) để biểu diễn các thuộc tính và giá trị sử dụng mô hình dữ liệu RDF
 - Từ khóa *@context*
 - ngữ cảnh còn có thể được thiết lập thông qua tham số API, hoặc thông qua 1 liên kết trong tiêu đề của phản hồi HTTP

Mỗi từ khóa *@context*, *@id*, *@type*, v.v. có ý nghĩa riêng.

```
{
  "@context": "http://schema.org",
  "@id": "http://www.janedoe.com",
  "@type": "Person",
  "name": "Jane Doe",
  "image": "http://localhost:9393/examples/schema.org/janedoe.jpg",
  "colleagues": [
    "http://www.xyz.edu/students/alicejones",
    "http://www.xyz.edu/students/bobsmith"
  ]
}
```

Dữ liệu trong HTML

```
<script type="application/ld+json">
{
  "@context": "https://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
</script>
```

- Dữ liệu liên kết từ nhiều trang có thể được tích hợp lại thành 1 đồ thị.
- Google khuyến khích tăng cường chèn JSON-LD vào trang Web
- Để xuất đúng dữ liệu từ trang Web có thể cần thực thi mã cập nhật cấu trúc DOM của tài liệu.

Nội dung

2.1. Căn bản XML

2.2. Xử lý dữ liệu XML

2.3. Căn bản JSON

2.4. Xử lý dữ liệu JSON



Môi trường lập trình

Javascript

jsonld.js 1.1

jsonld-streaming-parser.js 1.1

jsonld-streaming-serializer.js 1.1

rdf-parse.js 1.1

Java

Titanium 1.1

JSONLD-JAVA 1.0

Python

PyLD 1.1

RDFLib-jsonld 1.0

Go

JSON-goLD 1.1

PHP

php-json-ld 1.0

JsonLD 1.0

JSON-LD trong Python với pyld

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
semweb json-jd-demo.py
json-jd-demo.py x
1 from pyld import jsonld
2 import json
3 doc = { doc:
4     "http://schema.org/name": "Manu Sporny",
5     "http://schema.org/url": {"@id": "http://manu.sporny.org/"},
6     "http://schema.org/image": {"@id": "http://manu.sporny.org/images/manu.png"}
7 }
8 context = { context:
9     "name": "http://schema.org/name",
10    "homepage": {"@id": "http://schema.org/url", "@type": "@id"},
11    "image": {"@id": "http://schema.org/image", "@type": "@id"}
12 }
13 # compact a document according to a particular context
14 # see: https://json-ld.org/spec/latest/json-ld/#compacted-document-form
15 compacted = jsonld.compact(doc, context) compacted:
16 print(json.dumps(compacted, indent=2))
17
18 # compact using URLs
19 jsonld.compact('http://example.org/doc', 'http://example.org/context')
20
21 # expand a document, removing its context
22 # see: https://json-ld.org/spec/latest/json-ld/#expanded-document-form
23 expanded = jsonld.expand(compacted)
24
25 print(json.dumps(expanded, indent=2))
26
27 # expand using URLs
28 jsonld.expand('http://example.org/doc')
29
30 # flatten a document
31 # see: https://json-ld.org/spec/latest/json-ld/#flattened-document-form
32 flattened = jsonld.flatten(doc)
```

```
Debug - json-jd-demo
Console
{
  "@context": {
    "name": "http://schema.org/name",
    "homepage": {
      "@id": "http://schema.org/url",
      "@type": "@id"
    },
    "image": {
      "@id": "http://schema.org/image",
      "@type": "@id"
    }
  },
  "image": "http://manu.sporny.org/images/manu.png",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/"
}
```

Các giải thuật lỗi

- Mở rộng - Expand, áp dụng ngữ cảnh để tạo tài liệu với các IRIs và các giá trị tuyệt đối.
- Vắn tắt - Compact, sử dụng ngữ cảnh để thay thế các IRIs bằng các từ khóa và biểu diễn các giá trị như các chuỗi nếu có thể.
- Làm phẳng - Flatten, loại bỏ những lồng ghép bên trong, sử dụng nút rỗng để kết nối các nút
- Tới RDF - To RDF, biến đổi JSON-LD thành RDF
- Từ RDF - From RDF, biến đổi RDF thành JSON-LD
- Khung - Frame, áp dụng 1 văn bản khung cho 1 tài liệu JSON-LD phẳng.

Ví dụ 2.9. JSON-LD: Mở rộng

```
{
  "@context": "http://drupal.example.org/
context.jsonld",
  "@id": "http://directory.occupy.net/occupation/al/
occupy-prishtina",
  "@type": ["ows:Occupation",
"schema:Organization"],
  "foaf:name": "Occupy Prishtina",
  "schema:name": "Occupy Prishtina",
  "dc:date": "2012-02-12T21:18:08-05:00",
  "dc:created": "2012-02-12T21:18:08-05:00",
  "dc:modified": "2012-02-12T21:57:52-05:00",
  "sioc:num_replies": 0,
  "schema:foundingDate":
"2011-10-15T00:00:00-04:00",
  "ows:twitter_account": "@OccupyPrishtina",
  "foaf:mbox": "",
  "schema:telephone": "",
  "foaf:phone": ""
}

[
  {
    "@id": "http://directory.occupy.net/occupation/al/occupy-prishtina",
    "@type": ["http://vocab.occupy.net/ows#Occupation", "http://
schema.org/Organization"],
    "http://purl.org/dc/terms/created": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2012-02-12T21:18:08-05:00"
      }
    ],
    "http://purl.org/dc/terms/date": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2012-02-12T21:18:08-05:00"
      }
    ],
    "http://purl.org/dc/terms/modified": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2012-02-12T21:57:52-05:00"
      }
    ],
    "http://rdfs.org/sioc/ns#num_replies": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#integer",
        "@value": "0"
      }
    ],
    "http://schema.org/foundingDate": [
      {
        "@type": "http://www.w3.org/2001/XMLSchema#date",
        "@value": "2011-10-15T00:00:00-04:00"
      }
    ],
    "http://schema.org/name": ["Occupy Prishtina"],
    "http://schema.org/telephone": [""],
    "http://vocab.occupy.net/ows#twitter_account": ["@OccupyPrishtina"],
    "http://xmlns.com/foaf/0.1/mbox": [""],
    "http://xmlns.com/foaf/0.1/name": ["Occupy Prishtina"],
    "http://xmlns.com/foaf/0.1/phone": [""],
  }
]
```


Ví dụ 2.10. JSON-LD: áp dụng khung

```
{
  "@context": {
    "@vocab": "http://example.org/",
    "contains": {"@type": "@id"}
  },
  "@graph": [{
    "@id": "http://example.org/library",
    "@type": "Library",
    "location": "Athens",
    "contains": "http://example.org/library/the-republic"
  }, {
    "@id": "http://example.org/library/the-republic",
    "@type": "Book",
    "creator": "Plato",
    "title": "The Republic",
    "contains": "http://example.org/library/the-republic#introduction"
  }, {
    "@id": "http://example.org/library/the-republic#introduction",
    "@type": "Chapter",
    "description": "An introductory chapter on The Republic.",
    "title": "The Introduction"
  }
}]
}
```

Dữ liệu phẳng

```
{
  "@context": {"@vocab": "http://example.org/"},
  "@type": "Library",
  "contains": {
    "@type": "Book",
    "contains": {
      "@type": "Chapter"
    }
  }
}
```

Khung

```
{
  "@context": {"@vocab": "http://example.org/"},
  "@id": "http://example.org/library",
  "@type": "Library",
  "location": "Athens",
  "contains": {
    "@id": "http://example.org/library/the-republic",
    "@type": "Book",
    "creator": "Plato",
    "title": "The Republic",
    "contains": {
      "@id": "http://example.org/library/the-republic#introduction",
      "@type": "Chapter",
      "description": "An introductory chapter on The Republic.",
      "title": "The Introduction"
    }
  }
}
```

Sau khi áp dụng khung

