

BÀI 6. JSON VÀ DOM

Nội dung:

I. JSON	1
1. Tổng quan	1
2. Sử dụng JSON	2
3. Đặc điểm của JSON.....	2
4. Cú pháp JSON	3
5. Kiểu dữ liệu của JSON	4
6. So sánh JSON và XML	4
7. Các hàm JSON.....	6
8. Một số hạn chế của JSON	6
9. Json trong JavaScript.....	6
10. Json trong C#	10
II. DOM	13
1. Giới thiệu về DOM.....	13
2. DOM là gì.....	13
3. Xử lý dữ liệu XML dùng mô hình DOM	13
4. Mô hình đối tượng tài liệu – DOM.....	14
5. Không gian tên System.Xml.....	15
6. Một số các class trong không gian tên System.Xml.....	17

I. JSON

1. Tổng quan

JSON (JavaScript Object Notation) là một tiêu chuẩn mở dựa trên văn bản nhẹ được thiết kế cho trao đổi dữ liệu có thể đọc được của con người. Các quy ước được sử dụng bởi JSON được các lập trình viên biết đến, bao gồm C, C ++, Java, Python, Perl, v.v.

- ✓ JSON là viết tắt của JavaScript Object Notation.
- ✓ Định dạng do Douglas Crockford chỉ định.
- ✓ Nó được thiết kế để trao đổi dữ liệu có thể đọc được của con người.
- ✓ Nó đã được mở rộng từ ngôn ngữ kịch bản JavaScript.
- ✓ Phần mở rộng của tên tệp là .json.
- ✓ Kiểu JSON Internet Media là application / json.
- ✓ Định danh Loại thống nhất là public.json.

2. Sử dụng JSON

- ✓ JSON được sử dụng trong khi viết các ứng dụng dựa trên JavaScript bao gồm các tiện ích mở rộng của trình duyệt và các trang web.
- ✓ Định dạng JSON được sử dụng để tuần tự hóa và truyền dữ liệu có cấu trúc kết nối mạng.
- ✓ JSON chủ yếu được sử dụng để truyền dữ liệu giữa máy chủ và các ứng dụng web.
- ✓ Các dịch vụ web và API sử dụng định dạng JSON để cung cấp dữ liệu công khai.
- ✓ JSON có thể được sử dụng với các ngôn ngữ lập trình hiện đại.

3. Đặc điểm của JSON

- ✓ JSON dễ dàng đọc và viết
- ✓ JSON là một định dạng trao đổi dựa trên text plain.
- ✓ JSON độc lập với ngôn ngữ.

*Một ví dụ JSON

Ví dụ sau cho thấy cách sử dụng JSON để lưu trữ thông tin liên quan đến cuốn sách (book):

```
{
  "book":
    [
      {
        "id": "01",
        "language": "Java",
        "edition": "third",
        "author": "Herbert Schildt"
      },
      {
        "id": "07",
        "language": "C++",
        "edition": "second",
        "author": "E.Balagurusamy"
      }
    ]
}
```

File html có nội dung sau:

```
<html>
<head>
  <title>JSON example</title>

  <script language="javascript">

    var object1 = { "language": "Java", "author": "herbert schildt" };
    document.write("<h1>JSON with JavaScript example</h1>");
    document.write("<br />Object 1:");
    document.write("<h3>Language = " + object1.language + "</h3>");
    document.write("<h3>Author = " + object1.author + "</h3>");
```

```

var object2 = { "language": "C++", "author": "E-Balagurusamy" };
document.write("<br />Object 2:");
document.write("<h3>Language = " + object2.language + "</h3>");
document.write("<h3>Author = " + object2.author + "</h3>");

document.write("<hr />");
document.write(object2.language + " programming language can be studied " +
    "from book written by " + object2.author);
document.write("<hr />");

</script>

</head>

<body>
</body>

</html>

```

Kết quả chạy chạy trên trình duyệt:

JSON with JavaScript example

Object 1:
Language = Java
Author = herbert schildt

Object 2:
Language = C++
Author = E-Balagurusamy

C++ programming language can be studied from book written by E-Balagurusamy

4. Cú pháp JSON

Cú pháp JSON về cơ bản được coi là tập hợp con của cú pháp JavaScript; nó bao gồm những điều sau:

- ✓ Dữ liệu được biểu diễn dưới dạng khóa / giá trị.
- ✓ Dấu ngoặc nhọn chứa các đối tượng và mỗi tên được theo sau bởi ':' (dấu hai chấm), tên / giá trị
- ✓ Các cặp được phân tách bằng dấu, (dấu phẩy).
- ✓ Dấu ngoặc vuông chứa các mảng và các giá trị được phân tách bằng dấu, (dấu phẩy).

Dưới đây là một ví dụ đơn giản:

```
{
  "book": [
    {
      "id": "01",
      "language": "Java",
      "edition": "third",
      "author": "Herbert Schildt"
    },
    {
      "id": "07",
      "language": "C++",
      "edition": "second",
      "author": "E.Balagurusamy"
    }
  ]
}
```

JSON hỗ trợ hai cấu trúc dữ liệu sau:

- Tập hợp các cặp tên / giá trị: Cấu trúc dữ liệu này được hỗ trợ bởi các ngôn ngữ lập trình.
- Danh sách có thứ tự các giá trị: Nó bao gồm mảng, danh sách, vector hoặc chuỗi, v.v.

5. Kiểu dữ liệu của JSON

Kiểu	Mô tả
Number	Kiểu dữ liệu số
String	Kiểu dữ liệu chuỗi ký tự
Boolean	Kiểu dữ liệu có 2 giá trị true và false
Array	Kiểu mảng
Value	Giá trị
Object	Kiểu đối tượng
null	Trống

6. So sánh JSON và XML

- Cả JSON và XML đều có thể được sử dụng để nhận dữ liệu từ máy chủ web.
- Các ví dụ JSON và XML sau đây đều xác định một đối tượng nhân viên, với một mảng gồm 3 nhân viên:

*Ví dụ JSON:

```
{
  "employees": [
    { "firstName": "John", "lastName": "Doe" },
    { "firstName": "Anna", "lastName": "Smith" },
    { "firstName": "Peter", "lastName": "Jones" }
  ]
}
```

*Ví dụ XML:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON giống như XML vì:

- Cả JSON và XML đều "tự mô tả" (con người có thể đọc được)
- Cả JSON và XML đều phân cấp (giá trị trong giá trị)
- Cả JSON và XML đều có thể được phân tích cú pháp và được sử dụng bởi rất nhiều ngôn ngữ lập trình
- Cả JSON và XML đều có thể được tìm nạp (fetch) bằng XMLHttpRequest

JSON không như XML vì:

- JSON không sử dụng thẻ kết thúc
- JSON ngắn hơn JSON đọc và viết nhanh hơn
- JSON có thể sử dụng mảng
- Sự khác biệt lớn nhất là: XML phải được phân tích cú pháp bằng trình phân tích cú pháp XML.
- JSON có thể được phân tích cú pháp bởi một hàm JavaScript tiêu chuẩn.

Tại sao JSON tốt hơn XML

- XML khó phân tích cú pháp hơn nhiều so với JSON.
- JSON được phân tích cú pháp thành một đối tượng JavaScript sẵn sàng sử dụng.
- Đối với các ứng dụng AJAX, JSON nhanh hơn và dễ dàng hơn so với XML:
- Sử dụng XML
 - Tìm nạp tài liệu XML
 - Sử dụng DOM XML để lặp qua tài liệu
 - Trích xuất các giá trị và lưu trữ trong các biến
- Sử dụng JSON
 - Tìm nạp một chuỗi JSON
 - JSON.Parse chuỗi JSON

7. Các hàm JSON

7.1. Hàm JSON.parse

Sử dụng hàm JavaScript JSON.parse () để chuyển đổi một chuỗi thành một đối tượng JavaScript:

Giả sử có chuỗi: '{ "name":"John", "age":30, "city":"New York"}'

Để chuyển chuỗi này thành đối tượng, chúng ta sử dụng:

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

7.2. JSON.stringify()

Sử dụng hàm JavaScript JSON.stringify () để chuyển đối tượng JavaScript thành một chuỗi.

```
var obj = { name: "John", age: 30, city: "New York" };  
var myJSON = JSON.stringify(obj);
```

8. Một số hạn chế của JSON

- Không có lược đồ. Một mặt, điều đó có nghĩa là chúng ta hoàn toàn có thể linh hoạt để trình bày dữ liệu theo bất kỳ cách nào chúng ta muốn. Mặt khác, điều đó có nghĩa là chúng ta có thể vô tình tạo ra dữ liệu dạng sai rất dễ dàng.
- Chỉ có một loại số: định dạng dấu phẩy động chính xác kép IEEE-754. Điều đó khá thú vị, nhưng nó chỉ đơn giản có nghĩa là bạn không thể tận dụng các loại số đa dạng và sắc thái có sẵn trong nhiều ngôn ngữ lập trình.
- Không có loại ngày. Thiếu sót này có nghĩa là các nhà phát triển phải sử dụng chuỗi biểu thị ngày, dẫn đến sự khác biệt về định dạng hoặc phải biểu diễn ngày ở dạng mili giây kể từ kỷ nguyên (ngày 1 tháng 1 năm 1970).
- Không có chú thích. Không thể chú thích dẫn đến có thể có sự hiểu nhầm trong đoạn mã.

9. Json trong JavaScript

Ví dụ 1: Chúng ta có xâu Json như sau:

```
{  
  "artists" : [  
    {  
      "artistname" : "Leonard Cohen",  
      "born" : "1934"  
    },  
    {  
      "artistname" : "Joe Satriani",  
      "born" : "1956"  
    },  
    {  
      "artistname" : "Snoop Dogg",  
      "born" : "1971"  
    }  
  ]  
}
```

Chúng ta sẽ sử dụng JavaScript để lấy dữ liệu JSON ở trên, định dạng bằng các thẻ HTML và xuất ra tài liệu HTML.

Đây là mã:

```
<!doctype html>
<title>Example</title>
<script>
function displayArtists() {

    // Our JSON data
    var data =
    {
        "artists" : [
            {
                "artistname" : "Leonard Cohen",
                "born" : "1934"
            },
            {
                "artistname" : "Joe Satriani",
                "born" : "1956"
            },
            {
                "artistname" : "Snoop Dogg",
                "born" : "1971"
            }
        ]
    }

    // Put the data into a variable and format with HTML tags
    var output = "<h1>Artists</h1>";
    output += "<ul>";

    // Loop through the artists
    for (var i in data.artists) {
        output += "<li>" + data.artists[i].artistname + " (Born: " +
data.artists[i].born + ")</li>";
    }

    output += "</ul>";

    // Output the data to the "artistList" element
    document.getElementById("artistList").innerHTML=output;
}

// Load the above function when the window loads
window.onload = displayArtists;
</script>

<!-- The output appears here -->
<div id="artistList"></div>
```

Kết quả hiển thị:

Artists

- Leonard Cohen (Born: 1934)
- Joe Satriani (Born: 1956)
- Snoop Dogg (Born: 1971)

Ví dụ 2: Sử dụng vòng lặp lồng nhau:

```
<!doctype html>
<title>Example</title>
<script>
function displayArtists() {

    // Our JSON data
    var data =
    {
        "artists" : [
            {
                "artistname" : "Deep Purple",
                "formed" : "1968",
                "albums" : [
                    {
                        "albumname" : "Machine Head",
                        "year" : "1972",
                        "genre" : "Rock"
                    },
                    {
                        "albumname" : "Stormbringer",
                        "year" : "1974",
                        "genre" : "Rock"
                    }
                ]
            },
            {
                "artistname" : "Joe Satriani",
                "born" : "1956",
                "albums" : [
                    {
                        "albumname" : "Flying in a Blue Dream",
                        "year" : "1989",
                        "genre" : "Instrumental Rock"
                    },
                    {
                        "albumname" : "The Extremist",
                        "year" : "1992",
                        "genre" : "Instrumental Rock"
                    },
                    {
                        "albumname" : "Shockwave Supernova",
                        "year" : "2015",
                        "genre" : "Instrumental Rock"
                    }
                ]
            }
        ]
    }
}
```



```

    },
    {
      "artistname" : "Snoop Dogg",
      "born" : "1971",
      "albums" : [
        {
          "albumname" : "Tha Doggfather",
          "year" : "1996",
          "genre" : "Gangsta Rap"
        },
        {
          "albumname" : "Snoopified",
          "year" : "2005",
          "genre" : "Gangsta Rap"
        }
      ]
    }
  ]
}

// Put the data into a variable and format with HTML tags
var output = "<h1>Artists</h1>";

// Loop through the artists
for (var i in data.artists) {
  output += "<h2>" + data.artists[i].artistname + "</h2>";
  output += "<ul>";

  // Loop through the albums for the current artist
  for (var j in data.artists[i].albums) {
    output += "<li>" + data.artists[i].albums[j].albumname;
  }
  output += "</ul>";
}

// Output the data to the "artistlist" element
document.getElementById("artistlist").innerHTML=output;
}

// Load the above function when the window loads
window.onload = displayArtists;
</script>

<!-- The output appears here -->
<div id="artistlist"></div>

```

Kết quả hiển thị:

Artists

Deep Purple

- Machine Head
- Stormbringer

Joe Satriani

- Flying in a Blue Dream
- The Extremist
- Shockwave Supernova

Snoop Dogg

- Tha Doggfather
- Snoopified

10. Json trong C#

Sử dụng thư viện using Newtonsoft.Json để Serialize và Deserialize.

Chúng ta có class Student như sau:

```
class Student
{
    public string sid{ get; set; }
    public string name{ get; set; }
    public int age{ get; set; }
    public string city{ get; set; }
    public Student()
    {
    }
    public Student(string sid, string name, int age, string city)
    {
        this.sid = sid;
        this.name = name;
        this.age = age;
        this.city = city;
    }
    public override string ToString()
    {
        return "sid: " + sid + ", name: " + name + ", age: " + age + ", city: " + city;
    }
}
```

a. Để Serialize (tuần tự hóa) một đối tượng Student, đoạn mã như sau:

```
static string SerializeAnObject1()
```

```

{
    var student = new Student
    {
        sid = "s01",
        name = "Mai Huong",
        age = 18,
        city = "Ha noi"
    };

    //var s = new Student("s01", "Van Thai", 20, "Hai phong");
    //or return JsonConvert.SerializeObject(s);
    return JsonConvert.SerializeObject(student, Formatting.Indented);
}

```

b. Để Deserialize Json thành đối tượng Student:

```

static Student DeserializeAObject()
{
    string strjSon = SerializeAnObject();
    Student s = JsonConvert.DeserializeObject<Student>(strjSon);
    return s;
}

```

c. Serialize một mảng các đối tượng

```

static string SerializeListOfObject()
{
    List<Student> sts = new List<Student>();

    sts.Add(new Student("s01", "Tran Long", 21, "Hai phong"));
    sts.Add(new Student("s02", "Nguyen Hang", 20, "Quang ninh"));
    sts.Add(new Student("s03", "Phan Mai", 20, "Ha noi"));
    return JsonConvert.SerializeObject(sts, Formatting.Indented);
}

```

d. Deserialize Json thành mảng đối tượng

```

static List<Student> DeserializeListObject()
{
    string strjSon = SerializeListOfObject();
    Console.WriteLine("=====\nJson return in method : " +
strjSon);
    List<Student> li
JsonConvert.DeserializeObject<List<Student>>(strjSon);
    return li;
}

```

e. Serial đối tượng và ghi vào file text

```

static void SerializeFileObject()
{
    // serialize JSON to a string and then write string to a file
    Student student = new Student("s01", "Ha Long Van", 25, "Hanoi");
    File.WriteAllText("student.json",
JsonConvert.SerializeObject(student));

    //or serialize JSON directly to a file
    using (StreamWriter file = File.CreateText("student2.json"))

```

```
        {  
            JsonSerializer serializer = new JsonSerializer();  
            serializer.Serialize(file, student);  
        }  
    }
```

f. Deserialize file text và ghi vào mảng đối tượng.

```
static void DeserializeFileObject()  
{  
    // read file into a string and deserialize JSON to a type  
    Student student =  
    JsonConvert.DeserializeObject<Student>(File.ReadAllText("student.json"));  
  
    // or deserialize JSON directly from a file  
    Student student2;  
    using (StreamReader file = File.OpenText("student.json"))  
    {  
        JsonSerializer serializer = new JsonSerializer();  
        student2 = (Student)serializer.Deserialize(file,  
        typeof(Student));  
    }  
    Console.WriteLine("Student: " + student);  
    Console.WriteLine("Student2: " + student2);  
}
```

Tham khảo:

https://www.tutorialspoint.com/json/json_tutorial.pdf

https://www.tutorialspoint.com/json/pdf/json_quick_guide.pdf

<https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html>

https://www.w3schools.com/js/js_json_intro.asp

https://www.quackit.com/json/tutorial/json_with_javascript.cfm

<https://www.newtonsoft.com/json/help/html/ConvertJsonToXml.htm>

<https://www.newtonsoft.com/json/help/html/SerializeCollection.htm>

II. DOM

1. Giới thiệu về DOM

- Mô hình đối tượng tài liệu (DOM) là biểu diễn dữ liệu của các đối tượng bao gồm cấu trúc và nội dung của tài liệu trên web.
- Chúng ta sẽ xem xét cách DOM đại diện cho một tài liệu HTML hoặc XML trong bộ nhớ và cách sử dụng các API để tạo nội dung web và ứng dụng.

2. DOM là gì

- Mô hình Đối tượng Tài liệu (DOM) là một giao diện lập trình cho các tài liệu HTML và XML. Nó đại diện cho trang để các chương trình có thể thay đổi cấu trúc, kiểu và nội dung của tài liệu. DOM đại diện cho tài liệu dưới dạng các nút và đối tượng. Bằng cách đó, các ngôn ngữ lập trình có thể kết nối với trang.
- Trang Web là một tài liệu. Tài liệu này có thể được hiển thị trong cửa sổ trình duyệt hoặc dưới dạng nguồn HTML. Nhưng là tài liệu giống nhau trong cả hai trường hợp. Mô hình Đối tượng Tài liệu (DOM) đại diện cho một tài liệu, có thể được thao tác. DOM là một đại diện hướng đối tượng của trang web, có thể được sửa đổi bằng ngôn ngữ kịch bản như JavaScript.

3. Xử lý dữ liệu XML dùng mô hình DOM

- Mô hình đối tượng tài liệu XML (DOM) xử lý dữ liệu XML như một tập hợp các đối tượng tiêu chuẩn và được sử dụng để xử lý dữ liệu XML trong bộ nhớ.
- Không gian tên System.Xml cung cấp các thao tác với các node trong tài liệu XML. Nó dựa trên Core DOM Level 1 của tổ chức World Wide Web Consortium (W3C) và các khuyến nghị về Core DOM Level 2.
- Sử dụng XmlDocument và các class liên quan với nó, chúng ta có thể tạo tài liệu XML, tải và truy cập dữ liệu, sửa đổi dữ liệu và lưu các thay đổi.
- **Trong bài học này, chúng ta sẽ tìm hiểu về:**
 - Mô hình đối tượng tài liệu XML (DOM)
 - Các kiểu của nút (node)
 - Cấu trúc phân cấp mô hình đối tượng tài liệu XML (DOM)
 - Không gian tên System.Xml
 - Một số class trong không gian tên System.Xml
 - Ánh xạ cấu trúc phân cấp đối tượng sang dữ liệu XML
 - Tạo tài liệu XML
 - Đọc tài liệu XML vào DOM
 - Thêm các nút vào một tài liệu XML
 - Sử dụng XPath
 - Xóa nút trong tài liệu XML
 - Sửa nút trong tài liệu XML
 - Lưu trữ tài liệu

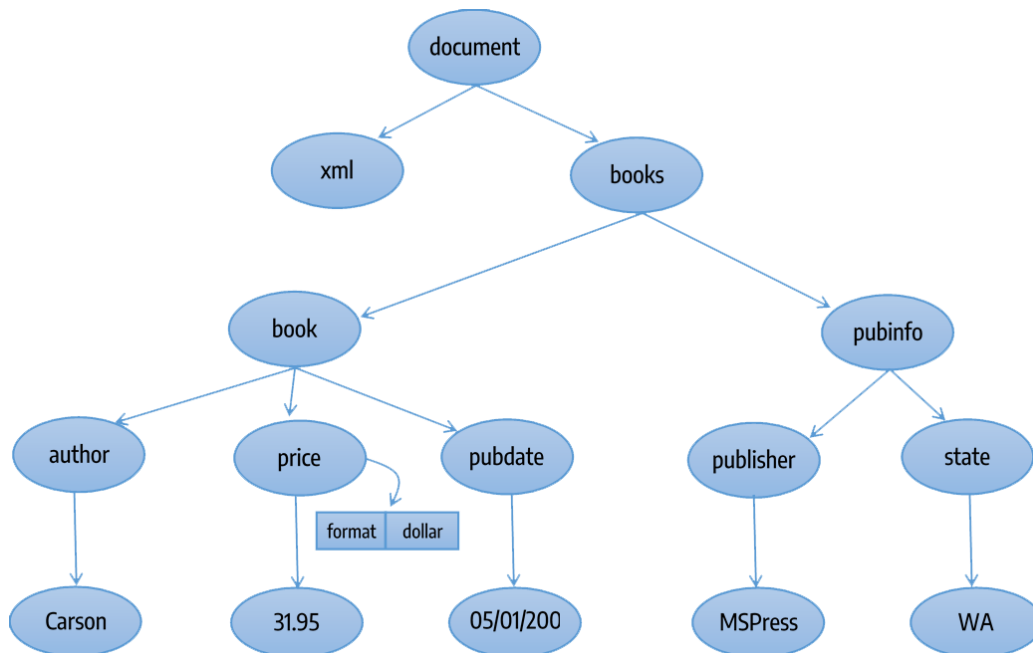
- Class XML Document Object Model (DOM) là một biểu diễn trong bộ nhớ của một tài liệu XML. DOM cho phép chúng ta đọc, thao tác và sửa đổi tài liệu XML theo chương trình.
- Class XmlReader cũng cho phép đọc XML; tuy nhiên, nó cung cấp quyền truy cập chỉ đọc, không lưu trong bộ nhớ cache, chỉ chuyển tiếp. Điều này có nghĩa là không có khả năng chỉnh sửa giá trị của một thuộc tính hoặc nội dung của một phần tử, hoặc khả năng chèn và loại bỏ các nút với XmlReader.
- Chỉnh sửa là chức năng chính của DOM. Đó là cách phổ biến và có cấu trúc mà dữ liệu XML được biểu diễn trong bộ nhớ, mặc dù dữ liệu XML thực tế được lưu trữ theo kiểu tuyến tính khi ở trong một tệp hoặc đến từ một đối tượng khác.

4. Mô hình đối tượng tài liệu – DOM

Chúng ta có tài liệu xml như sau:

```
<?xml version="1.0"?>
<books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
  <pubinfo>
    <publisher>MSPress</publisher>
    <state>WA</state>
  </pubinfo>
</books>
```

Tài liệu xml trên được mô hình hóa:



Trong cấu trúc tài liệu XML, mỗi vòng tròn trong hình minh họa này đại diện cho một nút, được gọi là đối tượng XmlNode.

5. Không gian tên System.Xml

Cung cấp các hỗ trợ tiêu chuẩn để xử lý XML.

Một số class của không gian tên System.Xml:

XmlAttribute	Đại diện cho một thuộc tính. Giá trị hợp lệ và giá trị mặc định cho thuộc tính được xác định trong định nghĩa loại tài liệu (DTD) hoặc lược đồ.
XmlAttributeCollection	Đại diện cho một tập hợp các thuộc tính có thể được truy cập bằng tên hoặc chỉ mục.
XmlBinaryReaderSession	Cho phép quản lý các chuỗi được tối ưu hóa theo cách năng động.
XmlBinaryWriterSession	Cho phép sử dụng từ điển động để nén các chuỗi phổ biến xuất hiện trong thư và duy trì trạng thái.
XmlCDataSection	Đại diện cho một phần CDATA.
XmlCharacterData	Cung cấp các phương thức thao tác văn bản được sử dụng bởi một số lớp.
XmlComment	Đại diện cho phần tử chú thích
XmlConvert	Mã hóa và giải mã các tên XML, đồng thời cung cấp các phương pháp để chuyển đổi giữa các kiểu CLR và các kiểu ngôn ngữ định nghĩa Lược đồ XML (XSD). Khi chuyển đổi kiểu dữ liệu, các giá trị trả về không phụ thuộc vào ngôn ngữ.
XmlDataDocument	Cho phép dữ liệu có cấu trúc được lưu trữ, truy xuất và thao tác thông qua tập dữ liệu quan hệ (a relational DataSet).
XmlDeclaration	Đại diện cho nút khai báo XML <? Xml version = '1.0' ...?>.
XmlDictionary	Thực thi một từ điển được sử dụng để tối ưu hóa việc triển khai trình đọc / ghi XML của Windows Communication Foundation (WCF).
XmlDictionaryReader	Một lớp trừu tượng mà Windows Communication Foundation (WCF) bắt nguồn từ XmlReader để thực hiện tuần tự hóa và giải mã.
XmlDocument	Đại diện cho một tài liệu XML. Có thể sử dụng lớp này để tải, xác thực, chỉnh sửa, thêm và định vị XML trong một tài liệu.
XmlDocumentType	Đại diện cho khai báo loại tài liệu.
XmlElement	Đại diện cho một phần tử.

XmlEntity	Đại diện cho một khai báo thực thể, chẳng hạn như <! ENTITY ...>.
XmlEntityReference	Đại diện cho một nút tham chiếu thực thể.
XmlException	Trả về thông tin chi tiết về ngoại lệ cuối cùng.
XmlLinkedNode	Lấy nút ngay trước hoặc sau nút này.
XmlNamedNodeMap	Đại diện cho một tập hợp các nút có thể được truy cập bằng tên hoặc chỉ mục.
XmlNamespaceManager	Giải quyết, thêm và xóa không gian tên vào một tập hợp và cung cấp quản lý phạm vi cho các không gian tên này.
XmlNode	Đại diện cho một nút trong tài liệu XML.
XmlNodeChangedEventArgs	Cung cấp dữ liệu cho các sự kiện NodeChanged, NodeInserted, NodeInserting, NodeRemoved và NodeRemoving.
XmlNodeList	Đại diện cho một tập hợp các nút có thứ tự.
XmlNodeReader	Đại diện cho một trình đọc cung cấp quyền truy cập chuyển tiếp nhanh, không được lưu trong bộ nhớ cache vào dữ liệu XML trong XmlNode.
XmlNotation	Đại diện cho một khai báo ký hiệu, chẳng hạn như <! NOTATION ...>.
XmlParserContext	Cung cấp tất cả thông tin ngữ cảnh được yêu cầu bởi XmlReader để phân tích cú pháp một đoạn XML.
XmlProcessingInstruction	Đại diện cho một hướng dẫn xử lý, XML định nghĩa để giữ thông tin cụ thể của bộ xử lý trong văn bản của tài liệu.
XmlQualifiedName	Đại diện cho một tên tiêu chuẩn XML.
XmlReader	Đại diện cho một trình đọc cung cấp quyền truy cập nhanh, không được lưu trong bộ nhớ cache, chỉ chuyển tiếp vào dữ liệu XML.
XmlReaderSettings	Chỉ định một tập hợp các tính năng để hỗ trợ trên đối tượng XmlReader được tạo bởi phương thức Create.
XmlResolver	Giải quyết các tài nguyên XML bên ngoài được đặt tên bằng Mã định danh tài nguyên đồng nhất (URI).
XmlText	Đại diện cho nội dung văn bản của một phần tử hoặc thuộc tính.
XmlTextReader	Đại diện cho một trình đọc cung cấp quyền truy cập nhanh, không lưu trong bộ nhớ cache, chỉ chuyển tiếp vào dữ liệu XML. <i>Bắt đầu với .NET Framework 2.0, chúng tôi khuyên bạn nên sử dụng lớp XmlReader thay thế.</i>
XmlTextWriter	Đại diện cho writer mà được cung cấp một cách nhanh chóng, không lưu trong bộ nhớ cache, chỉ chuyển tiếp để

	tạo luồng hoặc tệp chứa dữ liệu XML tuân theo Ngôn ngữ đánh dấu mở rộng W3C (XML) 1.0 và các không gian tên trong đề xuất XML. <i>Bắt đầu với .NET Framework 2.0, chúng tôi khuyên bạn nên sử dụng lớp XmlWriter thay thế.</i>
XmlWriter	Đại diện cho một trình writer cung cấp một cách nhanh chóng, không lưu trong bộ nhớ cache, chỉ chuyển tiếp để tạo các luồng hoặc tệp có chứa dữ liệu XML..
XmlWriterSettings	Chỉ định một tập hợp các tính năng để hỗ trợ trên đối tượng XmlWriter được tạo bởi phương thức Create.

6. Một số các class trong không gian tên System.Xml

6.1. Class XmlReader

Đại diện cho một trình đọc cung cấp quyền truy cập nhanh, không lưu trong bộ nhớ cache, chỉ chuyển tiếp vào dữ liệu XML.

Ví dụ 1.

```
static void ReaderDemo1()
{
    // Create an instance of XmlTextReader and call Read method to read the
    file
    XmlTextReader textReader = new XmlTextReader("Books.xml");
    textReader.Read();
    // If the node has value
    while (textReader.Read())
    {
        // Move to first element
        textReader.MoveToElement();
        Console.WriteLine("XmlTextReader Properties Test");
        Console.WriteLine("=====");
        // Read this element's properties and display them on console
        Console.WriteLine("Name:" + textReader.Name);
        Console.WriteLine("Base URI:" + textReader.BaseURI);
        Console.WriteLine("Local Name:" + textReader.LocalName);
        Console.WriteLine("Attribute          Count:" +
textReader.AttributeCount.ToString());
        Console.WriteLine("Depth:" + textReader.Depth.ToString());
        Console.WriteLine("Line          Number:" +
textReader.LineNumber.ToString());
        Console.WriteLine("Node Type:" + textReader.NodeType.ToString());
        Console.WriteLine("Attribute          Count:" +
textReader.Value.ToString());
    }
}
```

Ví dụ 2:

```
static void ReaderDemo2()
{
    var reader = XmlReader.Create("books.xml");
    reader.ReadToFollowing("book");

    do
    {

```

```

        reader.MoveToFirstAttribute();
        Console.WriteLine($"genre: {reader.Value}");

        reader.ReadToFollowing("title");
        Console.WriteLine($"title: {reader.ReadElementContentAsString()}");

        reader.ReadToFollowing("author");
        Console.WriteLine($"author:
{reader.ReadElementContentAsString()}");

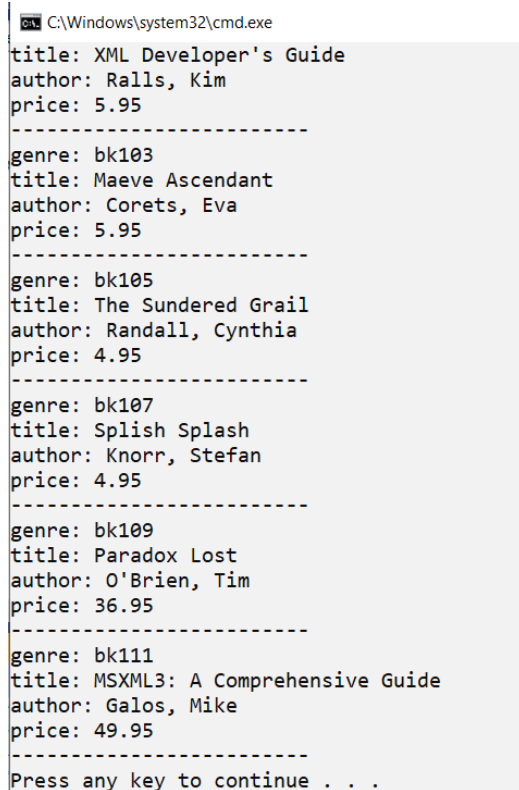
        reader.ReadToFollowing("price");
        Console.WriteLine($"price: {reader.ReadElementContentAsString()}");

        Console.WriteLine("-----");

    } while (reader.ReadToFollowing("book"));
}

```

Kết quả hiển thị:



```

C:\Windows\system32\cmd.exe
title: XML Developer's Guide
author: Ralls, Kim
price: 5.95
-----
genre: bk103
title: Maeve Ascendant
author: Corets, Eva
price: 5.95
-----
genre: bk105
title: The Sundered Grail
author: Randall, Cynthia
price: 4.95
-----
genre: bk107
title: Splish Splash
author: Knorr, Stefan
price: 4.95
-----
genre: bk109
title: Paradox Lost
author: O'Brien, Tim
price: 36.95
-----
genre: bk111
title: MSXML3: A Comprehensive Guide
author: Galos, Mike
price: 49.95
-----
Press any key to continue . . .

```

6.2. Class XmlTextReader

Đại diện cho một trình đọc cung cấp quyền truy cập nhanh, không lưu trong bộ nhớ cache, chỉ chuyển tiếp vào dữ liệu XML.

Bắt đầu với .NET Framework 2.0, nên sử dụng lớp XmlReader thay thế.

Ví dụ.

```

static void XmlTextReaderDemo()
{
    // Create an instance of XmlTextReader and call Read method to read the
    file
    XmlTextReader textReader = new XmlTextReader("Books.xml");
    textReader.Read();
}

```

```

    // If the node has value
    while (textReader.Read())
    {
        // Move to fist element
        textReader.MoveToElement();
        Console.WriteLine("XmlTextReader Properties Test");
        Console.WriteLine("=====");
        // Read this element's properties and display them on console
        Console.WriteLine("Name:" + textReader.Name);
        Console.WriteLine("Base URI:" + textReader.BaseURI);
        Console.WriteLine("Local Name:" + textReader.LocalName);
        Console.WriteLine("Attribute          Count:" +
textReader.AttributeCount.ToString());
        Console.WriteLine("Depth:" + textReader.Depth.ToString());
        Console.WriteLine("Line          Number:" +
textReader.LineNumber.ToString());
        Console.WriteLine("Node Type:" + textReader.NodeType.ToString());
        Console.WriteLine("Attribute          Count:" +
textReader.Value.ToString());
    }
  }
}

```

6.3. Class XmlWriter

Đại diện cho một trình viết cung cấp một cách nhanh chóng, không lưu trong bộ nhớ cache, chỉ chuyển tiếp để tạo các luồng hoặc tệp có chứa dữ liệu XML.

Ví dụ 1.

```

static void XmlWriterDemo1 ()
{
    using (XmlWriter writer = XmlWriter.Create("books1.xml"))
    {
        writer.WriteStartElement("book");
        writer.WriteElementString("title", "Graphics Programming using
GDI+");
        writer.WriteElementString("author", "Mahesh Chand");
        writer.WriteElementString("publisher", "Addison-Wesley");
        writer.WriteElementString("price", "64.95");
        writer.WriteEndElement();
        writer.Flush();
    }
}

```

Ví dụ 2.

```

static void XmlWriterDemo2()
{
    XmlWriterSettings settings = new XmlWriterSettings();
    settings.Indent = true;
    settings.IndentChars = ("  ");
    settings.CloseOutput = true;
    // settings.OmitXmlDeclaration = true;
    using (XmlWriter writer = XmlWriter.Create("books2.xml", settings))
    {
        writer.WriteComment("This is a book sample XML");
        writer.WriteStartElement("book");
        // Write ISBN attribute
        writer.WriteAttributeString("ISBN", "9831123212");
    }
}

```

```

        // Write year attribute
        writer.WriteStringAttribute("yearpublished", "2002");
        // Write title
        writer.WriteStringElement("author", "Mahesh Chand");
        // Write author
        writer.WriteStringElement("title", "Visual C# Programming");
        // Write price
        writer.WriteStringElement("price", "44.95");
        // Root element - end tag
        writer.WriteEndElement();
        // End Document
        writer.WriteEndDocument();
        // Flush it
        writer.Flush();
    }
}

```

Kết quả, chúng ta có file book2.xml như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<!--This is a book sample XML-->
<book ISBN="9831123212" yearpublished="2002">
  <author>Mahesh Chand</author>
  <title>Visual C# Programming</title>
  <price>44.95</price>
</book>

```

Ví dụ 3.

```

static void WriterDemo2()
{
    var sts = new XmlWriterSettings()
    {
        Indent = true,
    };

    var writer = XmlWriter.Create("Products.xml", sts);

    var products = new List<Product>
    {
        new Product(1, "Product A", 12.2M),
        new Product(2, "Product B", 11.3M),
        new Product(3, "Product C", 9M),
        new Product(4, "Product D", 5.6M),
        new Product(5, "Product E", 11.7M)
    };

    writer.WriteStartDocument();
    writer.WriteStartElement("products");

    foreach (var product in products)
    {
        writer.WriteStartElement("product");

        writer.WriteStartElement("id");
        writer.WriteValue(product.id);
    }
}

```

```

        writer.WriteEndElement();

        writer.WriteStartElement("name");
        writer.WriteValue(product.name);
        writer.WriteEndElement();

        writer.WriteStartElement("price");
        writer.WriteValue(product.price);
        writer.WriteEndElement();

        writer.WriteEndElement(); // end product
    }

    writer.WriteEndElement();
    writer.WriteEndDocument();
    writer.Flush();

    Console.WriteLine("XML document created");

    // record Product(int id, string name, decimal price);
}

```

Kết quả, chúng ta có file book2.xml như sau:

```

<?xml version="1.0" encoding="utf-8"?>
<products>
  <product>
    <id>1</id>
    <name>Product A</name>
    <price>12.2</price>
  </product>
  <product>
    <id>2</id>
    <name>Product B</name>
    <price>11.3</price>
  </product>
  <product>
    <id>3</id>
    <name>Product C</name>
    <price>9</price>
  </product>
  <product>
    <id>4</id>
    <name>Product D</name>
    <price>5.6</price>
  </product>
  <product>
    <id>5</id>
    <name>Product E</name>
    <price>11.7</price>
  </product>
</products>

```

6.4. Class XmlTextWriter

Đại diện cho writer mà cung cấp một cách nhanh chóng, không lưu trong bộ nhớ cache, chỉ chuyển tiếp để tạo luồng hoặc tệp chứa dữ liệu XML tuân theo Ngôn ngữ đánh dấu mở rộng W3C (XML) 1.0 và các không gian tên trong đề xuất XML.

Bắt đầu với .NET Framework 2.0, nên sử dụng lớp XmlWriter thay thế.

```
using System;
using System.IO;
using System.Xml;
public class Sample
{
    public static void Main()
    {
        XmlTextWriter writer = new XmlTextWriter("myMedia.xml", null);

        //Use automatic indentation for readability.
        writer.Formatting = Formatting.Indented;

        //Write the root element
        writer.WriteStartElement("items");

        //Start an element
        writer.WriteStartElement("item");

        //Add an attribute to the previously created element
        writer.WriteAttributeString("rating", "R");

        //add sub-elements
        writer.WriteElementString("title", "The Matrix");
        writer.WriteElementString("format", "DVD");

        //End the item element
        writer.WriteEndElement(); // end item

        //Write some white space between nodes
        writer.WriteWhitespace("\n");

        //Write a second element using raw string data
        writer.WriteRaw("<item>" +
            "<title>BloodWake</title>" +
            "<format>XBox</format>" +
            "</item>");

        //Write a third element with formatting in the string
        writer.WriteRaw("\n <item>\n" +
            "    <title>Unreal Tournament 2003</title>\n" +
            "    <format>CD</format>\n" +
            " </item>\n");
    }
}
```

```
// end the root element
writer.WriteFullEndElement();

//Write the XML to file and close the writer
writer.Close();
}
}
```

Kết quả, file myMedia.xml nhận được:

```
<items>
  <item rating="R">
    <title>The Matrix</title>
    <format>DVD</format>
  </item>
<item><title>BloodWake</title><format>XBox</format></item>
  <item>
    <title>Unreal Tournament 2003</title>
    <format>CD</format>
  </item>
</items>
```

6.5. Class XmlNode

Đại diện cho một nút (node) trong tài liệu XML.

***Đối tượng XmlNode là đối tượng cơ bản trong cây DOM.**

- Lớp XmlDocument, mở rộng XmlNode, hỗ trợ các phương thức để thực hiện các thao tác trên toàn bộ tài liệu (ví dụ: tải tài liệu vào bộ nhớ hoặc lưu XML vào một tệp).
- Ngoài ra, XmlDocument cung cấp các cách để xem và thao tác các nút trong toàn bộ tài liệu XML. Cả XmlNode và XmlDocument đều có các cải tiến về hiệu suất và khả năng sử dụng cũng như có các phương thức và thuộc tính để:
 - Truy cập và sửa đổi các nút cụ thể cho DOM, chẳng hạn như nút phần tử, nút tham chiếu thực thể, v.v.
 - Truy xuất toàn bộ nút, ngoài thông tin mà nút chứa, chẳng hạn như văn bản trong nút phần tử.
- Các đối tượng nút có một tập hợp các phương thức và thuộc tính, cũng như các đặc điểm cơ bản và được xác định rõ ràng. Một số đặc điểm này là:
 - Các nút có một nút cha duy nhất, một nút cha là một nút ngay phía trên chúng. Nút duy nhất không có nút cha là nút gốc tài liệu, vì nó là nút cấp cao nhất và chứa bản thân tài liệu và các đoạn tài liệu.
 - Hầu hết các nút có thể có nhiều nút con, là các nút ngay bên dưới chúng.
- Danh sách các loại nút có thể có các nút con:
 - Document

- DocumentFragment
- EntityReference
- Element
- Attribute

- Các nút **XmlDeclaration**, **Notation**, **Entity**, **CDATASection**, **Text**, **Comment**, **ProcessingInstruction** và **DocumentType** không có nút con.
- Các nút ở cùng cấp độ, được biểu thị trong sơ đồ bởi các nút <book> và nút <pubinfo>, được gọi là anh chị em.

*Thuộc tính trong DOM

- Một đặc điểm của DOM là cách nó xử lý các thuộc tính.
- Các thuộc tính không phải là các nút kiểu giống nhau, là một phần của các mối quan hệ cha, con và anh chị em.
- Các thuộc tính được coi là một thuộc tính của nút phần tử và được tạo thành từ một tên và một cặp giá trị.
- Ví dụ: nếu bạn có dữ liệu XML kiểu như format="dollar" được liên kết với phần tử price, format là tên thuộc tính, và giá trị là dollar. Để truy xuất thuộc tính format="dollar" của nút price, chúng ta gọi phương thức GetAttribute khi mà con trỏ hiện đang nằm ở nút phần tử price.

*Một số thuộc tính của class XmlNode

Attributes	Lấy ra XmlAttributeCollection chứa các thuộc tính của nút này.
ChildNodes	Lấy ra tất cả các nút con của nút.
FirstChild	Lấy ra phần tử con đầu tiên của nút.
HasChildNodes	Lấy ra một giá trị cho biết liệu nút này có bất kỳ nút con nào hay không.
InnerText	Lấy ra hoặc thiết lập giá trị text cho một nút
InnerXml	Lấy hoặc đặt đánh dấu đại diện cho các nút con của nút này.
IsReadOnly	Lấy ra một giá trị cho biết liệu nút có ở chế độ chỉ đọc hay không.
Item[String, String]	Lấy ra phần tử con đầu tiên với LocalName và NamespaceURI được chỉ định.
Item[String]	Lấy ra phần tử con đầu tiên với Tên được chỉ định.
LastChild	Lấy ra con cuối cùng của nút.
Name	Lấy ra tên tiêu chuẩn của nút, khi được ghi đè trong một lớp dẫn xuất.

NextSibling	Lấy ra nút ngay sau nút này.
NodeType	Lấy ra kiểu của nút hiện tại, khi được ghi đè trong một lớp dẫn xuất.
ParentNode	Lấy ra cha mẹ của nút này (đối với các nút có thể có cha mẹ).
PreviousSibling	Lấy nút ngay trước nút này.
Value	Lấy ra hoặc đặt giá trị của nút.

***Một số phương thức của class XmlNode:**

AppendChild(XmlNode)	Thêm nút được chỉ định vào cuối danh sách các nút con của nút này.
Clone()	Tạo một bản sao của nút này.
CloneNode(Boolean)	Tạo một bản sao của nút, khi được ghi đè trong một lớp dẫn xuất.
CreateNavigator()	Tạo XPathNavigator để điều hướng đối tượng này.
Equals(Object)	Xác định xem đối tượng được chỉ định có bằng đối tượng hiện tại hay không.
GetType()	Lấy ra kiểu của phần tử hiện tại (kế thừa từ Object).
InsertAfter(XmlNode, XmlNode)	Chèn nút được chỉ định ngay sau nút tham chiếu được chỉ định.
InsertBefore(XmlNode, XmlNode)	Chèn nút được chỉ định ngay trước nút tham chiếu được chỉ định.
RemoveAll()	Loại bỏ tất cả các nút con và / hoặc các thuộc tính của nút hiện tại.
RemoveChild(XmlNode)	Loại bỏ nút con được chỉ định.
ReplaceChild(XmlNode, XmlNode)	Thay thế nút con oldChild bằng nút newChild.
SelectNodes(String)	Chọn danh sách các nút phù hợp với biểu thức XPath.
SelectNodes(String, XmlNamespaceManager)	Chọn danh sách các nút phù hợp với biểu thức XPath. Bắt kỳ tiền tố nào được tìm thấy trong biểu thức XPath đều được

	giải quyết bằng cách sử dụng XmlNamespaceManager được cung cấp.
SelectSingleNode(String)	Chọn XmlNode đầu tiên phù hợp với biểu thức XPath.
ToString()	Trả về một chuỗi đại diện cho các đối tượng hiện tại. (Kế thừa từ Object)

***Các giá trị của NodeType**

NodeType	Đối tượng	Mô tả
Document	XmlDocument	Vùng chứa tất cả các nút trong cây. Nó còn được gọi là gốc tài liệu.
DocumentFragment	XmlDocumentFragment	Là một vùng tạm thời, chứa một hoặc nhiều nút mà không có bất kỳ cấu trúc cây nào.
DocumentType	XmlDocumentType	Đại diện cho node <!DOCTYPE...>
EntityReference	XmlEntityReference	Đại diện cho văn bản tham chiếu thực thể không được mở rộng.
Element	XmlElement	Kiểu phần tử
Attr	XmlAttribute	Kiểu thuộc tính
ProcessingInstruction	XmlProcessingInstruction	Là nút chỉ thị
Comment	XmlComment	Là nút chú thích
Text	XmlText	Là nút văn bản thuộc về nút phần tử hay nút thuộc tính
CDATASection	XmlCDATASection	Là nút CDATA (văn bản không cần phân tích)
Entity	XmlEntity	Là khai báo <!ENTITY> trong tài liệu XML

Notation	XmlNotation	Đại diện cho một ký hiệu được khai báo trong DTD.
----------	-------------	---

*Cấu trúc phân cấp XML (DOM)



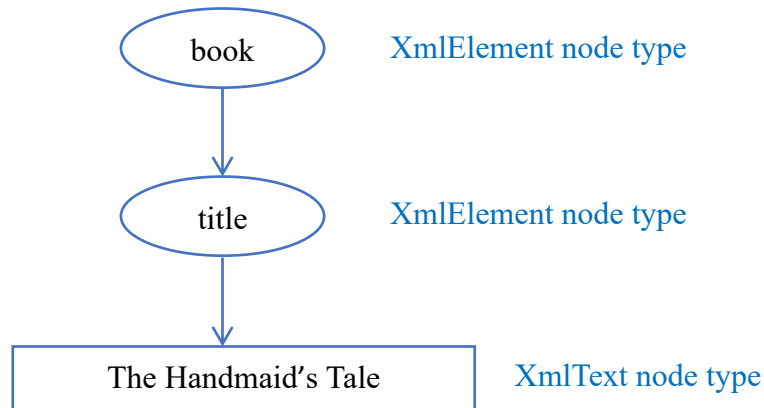
Các node sau đây không kế thừa từ **XmlNode**:

- XmlImplementation
- XmlNamedNodeMap
- XmlNodeList
- XmlNodeChangedEventArgs

*Ảnh xạ phân cấp đối tượng sang dữ liệu XML

- Khi một tài liệu XML nằm trong bộ nhớ, biểu diễn khái niệm là một cây. Để lập trình, bạn có một hệ thống phân cấp đối tượng để truy cập các nút của cây. Ví dụ sau cho bạn thấy cách nội dung XML trở thành các nút.

```
<book>
<title>The Handmaid's Tale</title>
</book>
```



6.6. Class XmlDocument

Là đại diện cho một tài liệu XML. Có thể sử dụng lớp này để tải, xác thực, chỉnh sửa, thêm và định vị XML trong một tài liệu.

Ví dụ, chúng ta có tài liệu Books.xml có nội dung sau:

```
<?xml version="1.0" encoding="utf-8"?>
<books xmlns="http://www.contoso.com/books">
  <book genre="novel" ISBN="1-861001-57-8" publicationdate="1823-01-28">
    <title>Pride And Prejudice</title>
    <price>24.95</price>
  </book>
  <book genre="novel" ISBN="1-861002-30-1" publicationdate="1985-01-01">
    <title>The Handmaid's Tale</title>
    <price>29.95</price>
  </book>
  <book genre="novel" ISBN="1-861001-45-3" publicationdate="1811-01-01">
    <title>Sense and Sensibility</title>
    <price>19.95</price>
  </book>
</books>
```

Đoạn mã sau nạp tài liệu này vào biến XmlDocument:

```
XmlDocument doc = new XmlDocument();
doc.PreserveWhitespace = true;
```

```
try { doc.Load("booksData.xml"); }
catch (System.IO.FileNotFoundException)
{
    doc.LoadXml("<?xml version='1.0' ?> \n" +
        "<books xmlns='http://www.contoso.com/books'> \n" +
        "    <book genre='novel' ISBN='1-861001-57-8' publicationdate='1823-01-28'> \n"
    +
        "        <title>Pride And Prejudice</title> \n" +
        "        <price>24.95</price> \n" +
        "    </book> \n" +
        "    <book genre='novel' ISBN='1-861002-30-1' publicationdate='1985-01-01'> \n"
    +
        "        <title>The Handmaid's Tale</title> \n" +
        "        <price>29.95</price> \n" +
        "    </book> \n" +
        "</books>");
}
```

Chúng ta có thể nạp tài liệu xml bằng xâu ký tự:

```
using System;
using System.IO;
using System.Xml;

public class Sample
{
    public static void Main()
    {
        //Create the XmlDocument.
        XmlDocument doc = new XmlDocument();
        doc.LoadXml("<?xml version='1.0' ?> " +
            "<book genre='novel' ISBN='1-861001-57-5'> " +
            "<title>Pride And Prejudice</title> " +
            "</book>");

        //Display the document element.
        Console.WriteLine(doc.DocumentElement.OuterXml);
    }
}
```

6.7. Class XElement

Đại diện cho một phần tử.

*Các thuộc tính của XElement

Attributes	Lấy ra XmlAttributeCollection chứa các thuộc tính của nút này.
ChildNodes	Lấy ra tất cả các nút con của nút.
FirstChild	Lấy ra phần tử con đầu tiên của nút.
HasChildNodes	Lấy ra một giá trị cho biết liệu nút này có bất kỳ nút con nào hay không.
InnerText	Lấy ra hoặc thiết lập giá trị text cho một nút
InnerXml	Lấy hoặc đặt đánh dấu đại diện cho các nút con của nút này.
IsReadOnly	Lấy ra một giá trị cho biết liệu nút có ở chế độ chỉ đọc hay không.
Item[String, String]	Lấy ra phần tử con đầu tiên với LocalName và NamespaceURI được chỉ định.
Item[String]	Lấy ra phần tử con đầu tiên với Tên được chỉ định.
LastChild	Lấy ra con cuối cùng của nút.
Name	Lấy ra tên tiêu chuẩn của nút, khi được ghi đè trong một lớp dẫn xuất.
NextSibling	Lấy ra nút ngay sau nút này.
NodeType	Lấy ra kiểu của nút hiện tại, khi được ghi đè trong một lớp dẫn xuất.
ParentNode	Lấy ra cha mẹ của nút này (đối với các nút có thể có cha mẹ).
PreviousSibling	Lấy nút ngay trước nút này.
PreviousText	Lấy ra nút văn bản ngay trước nút này.
Value	Lấy ra hoặc đặt giá trị của nút.

*Một số phương thức của XElement:

AppendChild(XmlNode)	Thêm nút được chỉ định vào cuối danh sách các nút con của nút này.
Clone()	Tạo một bản sao của nút này.

<u>CloneNode(Boolean)</u>	Tạo một bản sao của nút, khi được ghi đè trong một lớp dẫn xuất.
<u>CreateNavigator()</u>	Tạo XPathNavigator để điều hướng đối tượng này.
<u>Equals(Object)</u>	Xác định xem đối tượng được chỉ định có bằng đối tượng hiện tại hay không.
<u>GetType()</u>	Lấy ra kiểu của phần tử hiện tại (kế thừa từ Object).
<u>HasAttribute(String)</u>	Xác định xem nút hiện tại có thuộc tính với tên được chỉ định hay không.
<u>InsertAfter(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay sau nút tham chiếu được chỉ định.
<u>InsertBefore(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay trước nút tham chiếu được chỉ định.
<u>RemoveAll()</u>	Loại bỏ tất cả các nút con và / hoặc các thuộc tính của nút hiện tại.
<u>RemoveAllAttributes()</u>	Loại bỏ tất cả các thuộc tính được chỉ định khỏi phần tử. Các thuộc tính mặc định không bị xóa.
<u>RemoveAttribute(String)</u>	Xóa một thuộc tính theo tên.
<u>RemoveChild(XmlNode)</u>	Loại bỏ nút con được chỉ định.
<u>ReplaceChild(XmlNode, XmlNode)</u>	Thay thế nút con oldChild bằng nút newChild.
<u>SelectNodes(String)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath.
<u>SelectNodes(String, XmlNamespaceManager)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath. Bất kỳ tiền tố nào được tìm thấy trong biểu thức XPath đều được giải quyết bằng cách sử dụng XmlNamespaceManager được cung cấp.
<u>SelectSingleNode(String)</u>	Chọn XmlNode đầu tiên phù hợp với biểu thức XPath.
<u>SetAttribute(String, String)</u>	Đặt giá trị của thuộc tính với tên được chỉ định.
<u>ToString()</u>	Trả về một chuỗi đại diện cho các đối tượng hiện tại. (Kế thừa từ Object)

6.8. Class XMLText

Đại diện cho nội dung văn bản của một phần tử hoặc một thuộc tính.

*Các thuộc tính của XmlText

Attributes	Lấy ra XmlAttributeCollection chứa các thuộc tính của nút này.
ChildNodes	Lấy ra tất cả các nút con của nút.
FirstChild	Lấy ra phần tử con đầu tiên của nút.
HasChildNodes	Lấy ra một giá trị cho biết liệu nút này có bất kỳ nút con nào hay không.
InnerText	Lấy ra hoặc thiết lập giá trị text cho một nút
InnerXml	Lấy hoặc đặt đánh dấu đại diện cho các nút con của nút này.
IsReadOnly	Lấy ra một giá trị cho biết liệu nút có ở chế độ chỉ đọc hay không.
Item[String, String]	Lấy ra phần tử con đầu tiên với LocalName và NamespaceURI được chỉ định.
Item[String]	Lấy ra phần tử con đầu tiên với Tên được chỉ định.
LastChild	Lấy ra con cuối cùng của nút.
Length	Lấy ra độ dài của dữ liệu, tính bằng ký tự. (Được kế thừa từ XmlCharacterData)
Name	Lấy ra tên tiêu chuẩn của nút, khi được ghi đè trong một lớp dẫn xuất.
NextSibling	Lấy ra nút ngay sau nút này.
NodeType	Lấy ra kiểu của nút hiện tại, khi được ghi đè trong một lớp dẫn xuất.
ParentNode	Lấy ra cha mẹ của nút này (đối với các nút có thể có cha mẹ).
PreviousSibling	Lấy nút ngay trước nút này.
PreviousText	Lấy ra nút văn bản ngay trước nút này.
Value	Lấy ra hoặc đặt giá trị của nút.

***Một số phương thức của XmlText:**

<u>AppendChild(XmlNode)</u>	Thêm nút được chỉ định vào cuối danh sách các nút con của nút này.
<u>Clone()</u>	Tạo một bản sao của nút này.
<u>CloneNode(Boolean)</u>	Tạo một bản sao của nút, khi được ghi đè trong một lớp dẫn xuất.
<u>CreateNavigator()</u>	Tạo XPathNavigator để điều hướng đối tượng này.
<u>Equals(Object)</u>	Xác định xem đối tượng được chỉ định có bằng đối tượng hiện tại hay không.
<u>GetType()</u>	Lấy ra kiểu của phần tử hiện tại (kế thừa từ Object).
<u>InsertAfter(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay sau nút tham chiếu được chỉ định.
<u>InsertBefore(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay trước nút tham chiếu được chỉ định.
<u>RemoveAll()</u>	Loại bỏ tất cả các nút con và / hoặc các thuộc tính của nút hiện tại.
<u>RemoveChild(XmlNode)</u>	Loại bỏ nút con được chỉ định.
<u>ReplaceChild(XmlNode, XmlNode)</u>	Thay thế nút con oldChild bằng nút newChild.
<u>SelectNodes(String)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath.
<u>SelectNodes(String, XmlNamespaceManager)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath. Bất kỳ tiền tố nào được tìm thấy trong biểu thức XPath đều được giải quyết bằng cách sử dụng XmlNamespaceManager được cung cấp.
<u>SelectSingleNode(String)</u>	Chọn XmlNode đầu tiên phù hợp với biểu thức XPath.
<u>SplitText(Int32)</u>	Tách nút thành hai nút tại độ lệch được chỉ định, giữ cả hai trong cây như anh em ruột.
<u>Substring(Int32, Int32)</u>	Lấy một chuỗi con của chuỗi đầy đủ từ phạm vi được chỉ định. (Được kế thừa từ XmlCharacterData)
<u>SetAttribute(String, String)</u>	Đặt giá trị của thuộc tính với tên được chỉ định.
<u>ToString()</u>	Trả về một chuỗi đại diện cho các đối tượng hiện tại. (Kế thừa từ Object)

6.9. Class XMLAttribute

Đại diện cho một thuộc tính. Giá trị hợp lệ và giá trị mặc định cho thuộc tính được xác định trong định nghĩa loại tài liệu (DTD) hoặc lược đồ.

*Các thuộc tính của XmlAttribute

Attributes	Lấy ra XmlAttributeCollection chứa các thuộc tính của nút này.
ChildNodes	Lấy ra tất cả các nút con của nút.
FirstChild	Lấy ra phần tử con đầu tiên của nút.
HasChildNodes	Lấy ra một giá trị cho biết liệu nút này có bất kỳ nút con nào hay không.
InnerText	Lấy ra hoặc thiết lập giá trị text cho một nút
InnerXml	Lấy hoặc đặt đánh dấu đại diện cho các nút con của nút này.
IsReadOnly	Lấy ra một giá trị cho biết liệu nút có ở chế độ chỉ đọc hay không.
Item[String, String]	Lấy ra phần tử con đầu tiên với LocalName và NamespaceURI được chỉ định.
Item[String]	Lấy ra phần tử con đầu tiên với Tên được chỉ định.
LastChild	Lấy ra con cuối cùng của nút.
Name	Lấy ra tên tiêu chuẩn của nút, khi được ghi đè trong một lớp dẫn xuất.
NextSibling	Lấy ra nút ngay sau nút này.
NodeType	Lấy ra kiểu của nút hiện tại, khi được ghi đè trong một lớp dẫn xuất.
ParentNode	Lấy ra cha mẹ của nút này (đối với các nút có thể có cha mẹ).
PreviousSibling	Lấy nút ngay trước nút này.
PreviousText	Lấy ra nút văn bản ngay trước nút này.
Value	Lấy ra hoặc đặt giá trị của nút.

*Một số phương thức của XmlAttribute:

AppendChild(XmlNode)	Thêm nút được chỉ định vào cuối danh sách các nút con của nút này.
Clone()	Tạo một bản sao của nút này.

<u>CloneNode(Boolean)</u>	Tạo một bản sao của nút, khi được ghi đè trong một lớp dẫn xuất.
<u>CreateNavigator()</u>	Tạo XPathNavigator để điều hướng đối tượng này.
<u>Equals(Object)</u>	Xác định xem đối tượng được chỉ định có bằng đối tượng hiện tại hay không.
<u>GetType()</u>	Lấy ra kiểu của phần tử hiện tại (kế thừa từ Object).
<u>InsertAfter(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay sau nút tham chiếu được chỉ định.
<u>InsertBefore(XmlNode, XmlNode)</u>	Chèn nút được chỉ định ngay trước nút tham chiếu được chỉ định.
<u>RemoveAll()</u>	Loại bỏ tất cả các nút con và / hoặc các thuộc tính của nút hiện tại.
<u>RemoveChild(XmlNode)</u>	Loại bỏ nút con được chỉ định.
<u>ReplaceChild(XmlNode, XmlNode)</u>	Thay thế nút con oldChild bằng nút newChild.
<u>SelectNodes(String)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath.
<u>SelectNodes(String, XmlNamespaceManager)</u>	Chọn danh sách các nút phù hợp với biểu thức XPath. Bất kỳ tiền tố nào được tìm thấy trong biểu thức XPath đều được giải quyết bằng cách sử dụng XmlNamespaceManager được cung cấp.
<u>SelectSingleNode(String)</u>	Chọn XmlNode đầu tiên phù hợp với biểu thức XPath.
<u>ToString()</u>	Trả về một chuỗi đại diện cho các đối tượng hiện tại. (Kế thừa từ Object)
<u>WriteTo(XmlWriter)</u>	Lưu nút vào XmlWriter được chỉ định.

6.10. Class XmlNodeList

Đại diện cho một tập hợp các nút có thứ tự.

***Các thuộc tính của XmlNodeList**

<u>Count</u>	Lấy ra số lượng nút có trong XmlNodeList.
<u>ItemOf[Int32]</u>	Lấy ra một nút theo chỉ số.

***Các phương thức XmlNodeList:**

Equals(Object)	Xác định xem đối tượng được chỉ định có bằng đối tượng hiện tại hay không. (Kế thừa từ Object)
GetEnumerator()	Lấy ra một item trong tập hợp các nút. (Kế thừa từ Object)
GetType()	Lấy ra kiểu của đối tượng hiện tại. (Inherited from Object)
Item(Int32)	Truy xuất một nút tại chỉ mục đã cho.
MemberwiseClone()	Tạo một bản sao nông của Đối tượng hiện tại. (Kế thừa từ Object)
PrivateDisposeNodeList()	Xử lý tài nguyên trong danh sách nút một cách riêng tư.
ToString()	Trả về một chuỗi đại diện cho các đối tượng hiện tại. (Kế thừa từ Object)

Tài liệu tham khảo:

<https://docs.microsoft.com/en-us/dotnet/api/system.xml?view=net-5.0>