Journal of educational equipment: Education management, Volume 1, Issue 288 (May 2023)
ISSN 1859 - 0810

Phương pháp sử dụng dữ liệu XML cho ứng dụng WEB

Triệu Vĩnh Viêm*

** Khoa Công nghệ thông tin, Trường Đại học Bạc Liêu*

Received: 27/02/2023; Accepted: 06/03/2023; Published: 13/03/2023

Abstract: *In the era of digital transformation, data digitization is becoming increasingly diverse in order to store and transmit information between systems. Extracting this data for third-party systems to create separate storage systems is also crucial. JSON and XML are two popular formats for exchanging data between systems. Despite the popularity of JSON, XML is still used in essential operations such as Microsoft Navision ERP - Business Central, SOAP web services, distributed systems, and creating interface components for mobile applications. This article introduces an approach to extract electronic invoice data in XML format on a local machine to serve as a data source for web applications. This result is a reference for program developers who can extract component data from XML documents to create their own storage systems through web applications.*

Keywords: *Xml, web application, parse files, extract data, E-Invoice*

1. Đặt vấn đề

Từ nhu cầu quản lý dữ liệu mua vào/bán ra từ hóa đơn điện tử của mỗi doanh nghiệp nhằm phục vụ cho việc báo cáo thuế ngày càng trở nên thiết thực. Một số cá nhân và tổ chức có nghiệp vụ báo cáo thuế thay cho doanh nghiệp mong muốn tổ chức dữ liệu từ hóa đơn điện tử về kho dữ liệu riêng cho mỗi doanh nghiệp mà họ thực hiện báo cáo thuế thay. Từ đó, các cá nhân tổ chức này có nhu cầu trích lọc thông tin từ dữ liệu hóa đơn điện tử để quản lý. Từ nhu cầu này cho thấy việc trích xuất dữ liệu từ hóa đơn điện tử dạng XML là cần thiết cho các ứng dụng nghiệp vụ. Bài viết sẽ giới thiệu một tiếp cận lập trình duyệt qua tất cả dữ liệu hóa đơn dạng XML trong thư mục chỉ định, rồi sử dụng thư viện phân tích các tập tin dạng này để trích lọc dữ liệu và hiển thị lên ứng dụng web.

2. Nội dung nghiên cứu

2.1. Mô tả

Các định dạng trao đổi dữ liệu được phát triển từ định hướng đánh dấu và hiển thị để hỗ trợ thêm cho việc mã hóa siêu dữ liệu mô tả các thuộc tính cấu trúc của thông tin. Các yêu cầu để duy trì trao đổi dữ liệu của các kiến trúc ứng dụng đã dẫn đến sự phát triển của các định dạng trao đổi dữ liệu tiêu chuẩn. JSON và XML là hai định dạng trao đổi dữ liệu với các mục đích riêng.

2.1.1 Định dạng JSON

JSON (JavaScript Object Notation) được thiết kế để trở thành ngôn ngữ trao đổi dữ liệu mà con người có thể đọc được và máy tính dễ dàng phân tích và sử dụng. JSON được hỗ trợ trực tiếp bên trong JavaScript và phù hợp nhất cho các ứng dụng JavaScript; do đó cung cấp mức tăng hiệu suất đáng kể so với XML, vốn yêu cầu các thư viện bổ sung để truy xuất dữ liệu từ các đối tượng tài liệu (DOM). JSON được ước tính phân tích cú pháp nhanh hơn hàng trăm lần so với XML trong các trình duyệt hiện đại, nhưng bất chấp những tuyên bố về hiệu suất đáng chú ý của nó, các lập luận chống lại việc dùng JSON bao gồm việc thiếu hỗ trợ không gian tên, thiếu xác thực đầu vào và các hạn chế về khả năng mở rộng [1].

2.1.2. Định dạng XML

XML (eXtensible Markup Language) [2], hay ngôn ngữ đánh dấu mở rộng, là ngôn ngữ đánh dấu tiêu chuẩn và định dạng tệp để trao đổi dữ liệu. XML sử dụng đánh dấu phân cấp để lưu trữ và trao đổi dữ liệu, sử dụng các thẻ mở và thẻ đóng để phân định các thực thể nội dung dữ liệu với các thực thể khác trong cùng một tài liệu. Ta sẽ thường thấy mã XML được định dạng trong đó từng cấp độ của phần tử được thụt lề, giúp con người đọc tệp dễ dàng hơn trong khi không ảnh hưởng đến cách máy tính xử lý mã này. Dưới đây là một ví dụ về XML biểu diễn cho một phần tử ghi chú:

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

Hơn nữa, nếu muốn tổ chức nhiều phần tử ghi chú cho chương trình, ta có thể tổ chức lại nội dung của tập tin XML như sau:

```
<notes>
```

```
<note>
<to>Tomy</to>
<from>Ana</from>
<heading>Party to night</heading>
<body>At 9:00 PM, New Palace Restaurent</body>
</note>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
</notes>
```

Có thể hiểu XML là ngôn ngữ đánh dấu được phép định nghĩa thêm các thẻ theo nhu cầu ứng dụng bởi những nhà phát triển. Trong thiết kế mô hình ứng dụng hướng doanh nghiệp, định dạng XML cũng được sử dụng thường xuyên cho việc truyền tải và xử lý bởi các lời gọi thủ tục của mô hình khách - chủ [3]. Định dạng này còn được sử dụng bởi kiến trúc dịch vụ SOAP và thiết kế thành phần giao diện trong ứng dụng di động.

2.1.3. Tiếp cận trích xuất và trình bày dữ liệu

Có nhiều tiếp cận rút trích từ tập tin XML trong các nhóm phát triển ứng dụng khác nhau, trong bài viết này quan tâm đến các ứng dụng web có nhu cầu sử dụng dữ liệu dạng này. Đối với ứng dụng Web, có nhiều cách để đưa dữ liệu từ phía người sử dụng như nhận dữ liệu thông qua các thành phần thẻ HTML, hoặc ta cũng có thể nhập tự động nhiều dữ liệu đồng dạng thông qua đường dẫn chứa tài nguyên trên máy tính.

Trong chủ đề này, ta quy ước đã sẵn có các tập tin XML từ máy tính người dùng. Như vậy ta có thể dùng các kỹ thuật trong ngôn ngữ lập trình web như JS/TS để đọc vào. Một số tiếp cận để làm điều này đó là:

- Thư viện xml2json
- Thư viện xml2js
- Thư viện fast-xml-parser

Trong các phần tiếp theo sẽ đề cập việc sử dụng thư viện hiệu năng trong việc phân tích tài liệu xml là fast-xml-parser.

2.1.4. Duyệt tài nguyên từ máy người dùng

Xử lý các thư mục không phải là việc thường xuyên trong lập trình ứng dụng, nhưng đôi khi có trường hợp cần thiết để quét tất cả tài nguyên trên máy người dùng, chẳng hạn như muốn xử lý tất cả các hình ảnh (*.jpg, *.png, *.svg…) hoặc các tập tin có cấu trúc ở định dạng XML trong một thư mục bao hàm các thư mục cấp con.

Với File System Access API (tên gọi trước đây Native File System API, Writable Files API), nhà lập trình có thể dễ dàng mở các thư mục trong trình duyệt và quyết định xem có cần quyền ghi tài nguyên hay không.

API này cho phép tương tác với các tệp trên thiết bị cục bộ của người dùng hoặc trên hệ thống tệp mạng mà người dùng có thể truy cập. Chức năng cốt lõi của API này bao gồm đọc tệp, ghi hoặc lưu tệp và truy cập vào cấu trúc thư mục.

Hầu hết các tương tác với tệp và thư mục được thực hiện thông qua các Handle. Một lớp FileSystemHandle cha giúp định nghĩa hai lớp con: FileSystemFileHandle và FileSystemDirectoryHandle, tương ứng cho các tệp và thư mục.

Handles đại diện cho một tệp hoặc thư mục trên hệ thống của người dùng. Trước tiên, ta có thể truy cập chúng bằng cách hiển thị cho người dùng một bộ chọn tệp hoặc thư mục bằng cách sử dụng các phương thức như:

- window.showOpenFilePicker()
- window.showDirectoryPicker()

Một phần mã lệnh cho phép đọc tất cả tài nguyên XML ở chế độ "mode" có thể viết:

```
try {
    const handle = await showDirectoryPicker({ mode })
    return (await Promise.all(await getFiles(handle)))
        .filter((file) => file.name.endsWith('.xml'))
} catch (err: unknown) {
    if ((err as { name?: string }).name !== 'AbortError') {
        console.error((err as Error).name, (err as Error).message)
    }
    return null
}
```

Chế độ mode có thể ở dạng **'read'** hoặc **'readwrite'** tùy theo nhu cầu ứng dụng muốn truy cập ở quyền nào, nhưng cần được sự cho phép từ người dùng.

Thư viện File System Access API chỉ có thể được hỗ trợ vài trình duyệt ở phiên bản như công bố của nó. Để sử dụng được tính năng này trên trình duyệt ta cần thực hiện với giao thức HTTPS. Như vậy, nếu người dùng sử dụng trình duyệt Firefox/Safari hoặc

trang web chưa vận hành trên giao thức HTTPS thì người phát triển nên có tiếp cận thay thế là tạo ra thẻ input với kiểu dữ liệu tập tin như bên dưới:

```
new Promise((resolve) => {
    const input = document.createElement('input')
    input.type = 'file'
    input.webkitdirectory = true
    input.accept = 'text/xml'
    input.addEventListener('change', () => {
        const files = Array.from(input.files!)
            .filter((file) => file.name.endsWith('.xml'))
        resolve(files)
    })
    if ('showPicker' in HTMLInputElement.prototype) {
        input.showPicker()
    } else {
        input.click()
    }
})
```

Với cách vận hành duyệt tập tin trong môi trường ứng dụng web như trên người dùng có thể duyệt liên tục các tài nguyên chỉ định nào đó. Bằng cách này có thể duyệt tất cả các tài nguyên có phần mở rộng xml là dữ liệu các hóa đơn điện tử để đọc vào cho ứng dụng web.

Thư viện phân tích tập tin XML với ngôn ngữ JS/TS

Khi phát triển các dự án web trên môi trường quản lý gói của Node.js có nhiều thư viện phân tích tập tin XML như đã đề cập ở phần trước. Tuy nhiên, bài viết sử dụng Fast XML Parser bởi vì nó được cập nhật gần đây khi so với các thư viện khác và được sử dụng bởi nhiều tập đoàn/công ty lớn về công nghệ như: Microsoft, IBM, NASA, VMWARE… Fast XML Parser là một thư viện javascript để phân tích cú pháp XML thành đối tượng JS, JSON hoặc để xây dựng XML từ đối tượng JS mà không làm mất bất kỳ thông tin nào. Mục đích của thư viện này là cung cấp giải pháp nhanh chóng không có lỗi để xử lý định dạng dữ liệu XML trong javascript. Thư viện này cung cấp nhiều tùy chọn cho việc phân tích XML như: bỏ qua thuộc tính, chuyển đổi kiểu, tùy chỉnh định dạng xuất, hỗ trợ không gian tên và nhiều tính năng khác.

Trong dự án dễ dàng sử dụng nó như sau:

```
const { XMLParser, XMLBuilder, XMLValidator } = require("fast-xml-parser");
const parser = new XMLParser();
let jObj = parser.parse(XMLdata);
const builder = new XMLBuilder();
const xmlContent = builder.build(jObj);
```

Một phần mã lệnh cho phép ta xác định tính hợp lệ của tập tin xml ở dạng chuỗi:

```
const xmlString = e.target?.result as string
const isValidXml = XMLValidator.validate(xmlString)
if (!isValidXml) {
    console.error(`File ${file.name} is not a valid XML file.`)
    return
}
const options = {
    ignoreDeclaration: true,
    attributeNamePrefix: '@_'
}
const parser = new XMLParser(options)
const jsonObj = parser.parse(xmlString)
```

Với đối tượng jsonObj (đối tượng JSON) phân tích được, ta có thể truy xuất các dữ liệu thành phần bằng ngôn ngữ lập trình họ JS/TS. Từ đó, dễ dàng rút trích/điều chỉnh phân loại dữ liệu rồi lưu trữ vào cơ sở dữ liệu riêng của ứng dụng. Trường hợp lập trình động duyệt tài nguyên từ đường dẫn chỉ định có thể đọc nhiều tập tin hóa đơn, với mỗi hóa đơn là một đối tượng ta có thể đưa vào một mảng các hóa đơn trong JS/TS. Qua đó, dữ liệu từ mảng này có thể được dùng để hiển thị trên các thành phần giao diện.

3. Kết luận

Trong tương lai gần, tôi sẽ tiếp tục nâng cấp các chức năng khác của ứng dụng web này bằng cách như: gán nhãn phân loại hóa đơn dựa trên nhu cầu của người dùng, trích xuất và lưu trữ dữ liệu này vào cơ sở dữ liệu riêng của họ. Bên cạnh đó, tôi cũng sẽ nghiên cứu khả năng đọc một lượng lớn các tập tin XML vào ứng dụng web (hàng ngàn tệp) và đánh giá hiệu năng của nó.

Tài liệu tham khảo

[1] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study", 2009

[2] Extensible markup language (xml) 1.0 (fifth edition). W3C, 2008

[3] U. Hilger, "Article: Client/server with java and xml-rpc," 2005