

文件物件模型及其在 XML 文件處理之應用

淡江大學 資圖系
助理教授 林信成

Document Object Model and its Application on XML Document Processing

Sinn-Cheng Lin

Assistant Professor
Department of Information and Library Science,
Tamkang University,
Taipei, Taiwan, R.O.C.
E-Mail: sclin@mail.tku.edu.tw

Abstract

Document Object Model (DOM) is an application-programming interface that can be applied to process XML documents. It defines the logical structure, the accessing interfaces and the operation methods for the document. In the DOM, an original document is mapped to a tree structure. Therefore, the computer program can easily traverse the tree and manipulate the nodes in the tree. In this paper, the fundamental models, definitions and specifications of DOM are surveyed. Then we create an experimental system of DOM called XML On-Line Parser. The front-end of the system is built by the Web-based user interface for the XML document input and the parsed result output. On the other hand, the back-end of the system is built by an ASP program, which transforms the original document to DOM tree for document manipulation. This on-line system can be linked with a general-purpose web browser to check the well-formedness and the validity of the XML documents.

一、前言

(一) 研究背景與動機

資訊化社會所感困擾的不是資訊的匱乏，而是資訊的氾濫，尤其在網路普及的今日，如何有效的管理網路資源，以便在適當時機提供適當的資訊給適當的使用者，已經成為知識管理的重要議題。¹在眾多的解決方案之中，由全球資訊網協會（World Wide Web Consortium，簡稱 W3C）所倡議的可擴展標注語言（eXtensible Markup Language，簡稱 XML），²不但已被公認為主導「第二代 Web」（Second-Generation Web）的重要技術，也逐漸成為全球矚目的網路資源整合方

¹ Applehans, Globe, Laugero 著，馮國扶譯 知識管理 Any Time — 網上應用實作指南(Managing knowledge -A Practical Web-Based Approach)。台北市：和碩科技，民 88 年 8 月。

² 1996 年 7 月「XML 工作小組」（XML Working Group）在 W3C 的贊助下成立，當年 11 月提交 XML 初稿，並於 1998 年 1 月 10 日正式通過 XML 1.0 規範，成為 W3C 的一個建議標準（Recommendation）。

案。³XML 具有高度結構性、可擴展性和資料樣式分離原則，在資料的管理、交換上擁有極卓越之性能。XML 至少將帶來如下的效益：(1) 就知識管理而言，XML 由於具有良好的自我描述性，能夠有效的表達網路上的各種知識，為知識管理提供新的機制；(2) 就資訊檢索而言，XML 可以提供語意層次的搜尋，避免全面性的盲目搜索，進而提昇檢索結果的精確度 (Precision Rate)，這在網路資源氾濫成災的今日尤其重要；(3) 對於系統的開發而言，由於 XML 具備可擴展性、資料與樣式分離等特色，各個系統可根據自身的需求，對資料進行其他加值處理，這使得 Web 應用程式 (Web Application) 的發展更具彈性；(4) 由於 XML 文件的高度結構化，使得 XML 可以很容易的和目前已經發展成熟的各種資料庫管理系統 (DBMS) 進行資料交換，這意味著來自不同管道的資料將可以輕易的藉由 XML 加以整合。⁴

不過，要完全發揮 XML 之所長仍需要電腦程式的配合。XML 文件是一種基於文字模式的開放規格，雖然具備嚴謹的樹狀資料結構，但若要以電腦程式對其文件內容進行剖析、處理和操縱 (Manipulation)，仍需要藉助特定的演算法才行。幸好，樹 (Tree) 是一種極為普遍的資料結構，已經有許多現成的演算法被發展出來。⁵因此，任何應用程式都可以很容易的利用樹狀結構的演算法，對 XML 文件內容進行諸如讀取、修改、刪除、新增、搬移或走訪等動作。然而，若是能有一個介面標準直接支援 XML 樹狀結構的存取，將可簡化並縮短程式開發之流程。由 W3C 所推動的「文件物件模型」(Document Object Model, 簡稱 DOM) 便是一個這樣的介面標準。⁶不過，DOM 並非是針對 XML 量身定做的，它適用於 HTML 及 XML 文件。由於 DOM 是一個語言中立 (Language-Independent) 的應用程式介面 (Application Programming Interface, 簡稱 API)，目的在於建立一個跨平台的文件操作環境，因此，DOM 可以在不同的作業系統中，使用任何程式語言加以實現。在 DOM 的規範中，定義了文件的邏輯結構以及存取、處理、操縱文件的方法。藉由 DOM，應用程式可以輕易的建立文件，可以在文件結構中來回穿梭，可以新增、修改或刪除文件中的元素或內容。

(二) 研究目的與方法

本文宗旨在於深入探討 DOM，以作為 XML 系統實作之基礎。為了達到此一目標，除了研究相關文獻以深入瞭解 DOM 之定義、模型、結構、介面規範之外，尚透過系統實作、程式撰寫等方式，解析 DOM 之運作原理。全文架構如下：在前言之後，第二節為文件物件模型概述；第三節則說明 DOM 之樹狀結構；第四節闡釋 DOM 核心介面；第五節以 MSXML 剖析器為例，說明 DOM 之實作要點；第六節則依據 DOM 介面規範，建立一個 XML 線上剖析器實驗系統；最後，第七節為結論。

二、文件物件模型概述

DOM 規範依其複雜層度分為三個層級：Level 1、Level 2 和 Level 3。

³ Jon Bosak and Tim Bray, "XML and the Second-Generation Web", Scientific American, May 1999, also available at <<http://www.sciam.com/1999/0599issue/0599bosak.html>>.

⁴ 林信成，「XML 相關技術與下一代 Web 出版趨勢之研究」，教育資料與圖書館學，第 37 卷，第 2 期，頁 184-210，民 88 年。

⁵ Ellis Horowitz, "Fundamentals of data structures in Pascal," FREEMAN: San Francisco, 1989.

⁶ W3C Recommendation, "Document Object Model (DOM) Level 1 Specification Version 1.0", 1 October 1998, available at <<http://www.w3.org/TR/REC-DOM-Level-1>>.

(一) DOM Level 1

DOM 層級一 (DOM Level 1) 規格於 1998 年 10 月正式成為 W3C 的一個建議標準 (Recommendation),⁷ 其焦點集中在解決文件剖析的核心問題上, 定義了 HTML 和 XML 文件的結構模型, 提供文件巡航 (Navigation) 和操縱 (Manipulation) 等功能, 使得應用程式得以非常容易的存取文件內容 (新增、刪除、編修內容、編修屬性及文件類型 ...), 於是程式設計師便可藉以發展出適用於各種瀏覽器、伺服器及工作平台的應用程式; 程式設計師或許會使用不同的程式語言 (如 C/C++、Java、JavaScript、VB、VBScript ... 等), 但不需要去改變程式模型。⁸

(二) DOM Level 2

DOM 層級二 (DOM Level 2) 規格則於 2000 年 11 月 13 日成為建議標準, 其規格比 Level 1 複雜許多, 除了加強定義如何操縱及處理文件結構及內容的核心 (Core) 介面外⁹, 尚包含了文件的外觀 (Views)¹⁰、樣式表物件模型 (Style sheet object model)¹¹、事件模型 (Event model)¹²、走訪範圍 (Traversal Range)¹³ 等規範, 以便應用程式可以很容易的處理附加的樣式表、各種不同的外觀、事件以及允許在文件中四處走訪。

(三) DOM Level 3

DOM 層級三 (DOM Level 3) 規格至本文截稿時 (2000 年 12 月) 仍處於工作草案階段,¹⁴ 其內容主要包含了下列各項目:¹⁵

- ◆ 擴展 DOM 層級二之物件模型 (Object Model): 允許使用者存取鍵盤事件; 增加定義群組事件的能力。
- ◆ 新增內容模型 (Content Model) 和有效性 (Validation): 一個可以直接存取和修改文件內容之物件模型。
- ◆ 提供載入 (Load) 和儲存 (Save) 介面: 提供直接載入 XML 原始文件至 DOM 結構以及直接將 DOM 結構儲存為 XML 文件的程式介面。
- ◆ 內嵌式 (Embedded) 文件物件模型: 目前 Web 的發展趨勢正朝向以混合式標注語言方式寫作文件, 例如將 SVG 片段內嵌在 XHTML 文件中, 這使得 DOM 必須面對新的挑戰, 因為 DOM API 需要具備從混合的詞彙集中剖析文件結構與內容的能力。
- ◆ 適應 XML 核心功能改變的能力: DOM 既然是 XML 文件的 API, 一旦

⁷ 同註6。

⁸ W3C, "Document Object Model (DOM) Activity Statement," Last modified date: 2000/12/14, available at <<http://www.w3.org/DOM/Activity>>.

⁹ W3C Recommendation, "Document Object Model (DOM) Level 2 Core Specification Version 1.0," 13 November 2000, available at <<http://www.w3.org/TR/DOM-Level-2-Core>>.

¹⁰ W3C Recommendation, "Document Object Model (DOM) Level 2 Views Specification Version 1.0," 13 November 2000, available at <<http://www.w3.org/TR/DOM-Level-2-Views>>.

¹¹ W3C Recommendation, "Document Object Model (DOM) Level 2 Styles Specification Version 1.0," 13 November 2000, available at <<http://www.w3.org/TR/DOM-Level-2-Styles>>.

¹² W3C Recommendation, "Document Object Model (DOM) Level 2 Events Specification Version 1.0," 13 November 2000, available at <<http://www.w3.org/TR/DOM-Level-2-Events>>.

¹³ W3C Recommendation, "Document Object Model (DOM) Level 2 Traversal-Range Specification Version 1.0," 13 November 2000, available at <<http://www.w3.org/TR/DOM-Level-2-Traversal-Range>>.

¹⁴ W3C Working Draft, "Document Object Model (DOM) Level 3 Core Specification Version 1.0," 01 September 2000, available at <<http://www.w3.org/TR/DOM-Level-3-Core>>.

¹⁵ 同註8。

XML 發展出附加功能 (如 namespaces、XML Base 等), DOM API 也應能夠隨之調整其對應模型。

- ◆ XPath DOM : 使用 XPath 來查詢 DOM 結構樹的解決方案。

三、DOM 之樹狀結構

根據上述, 不論 Level 1、Level 2 或 Level 3, 實際上其根基為 DOM 核心規格 (DOM Core Specification), 其中定義了一個階層式物件模型。在此一物件模型中, 不論是文件、元素、屬性、實體、文字或其他在 XML 文件中出現的單元, 都被視為一個一個的節點 (Node), 而所有節點則被組織成一個樹狀結構 (Tree Structure)。¹⁶

XML 文件、剖析器、DOM 物件樹和應用程式之間的關係如圖 1 所示, 從圖中我們可以清楚的看到: XML 文件經由剖析器 (Parser) 剖析之後, 形成一個 DOM 樹狀結構, 此一樹狀結構將文件中的元素、屬性、資料 ... 等以階層式節點方式加以儲存, 應用程式則藉由存取樹中的節點而讀出資料或寫入資料。

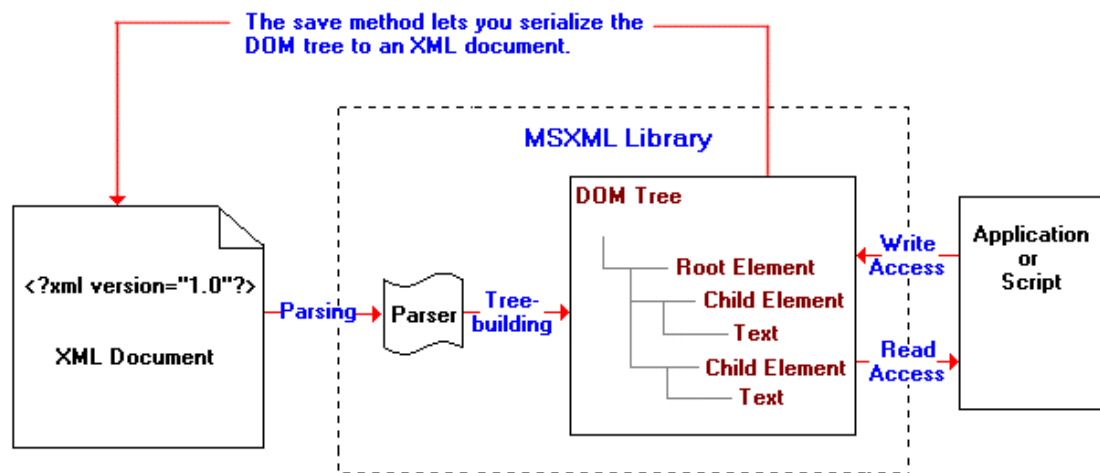


圖1 XML 文件之剖析與存取

資料來源: <http://msdn.microsoft.com/>

在 DOM 模型中總共定義了 12 種不同種類的節點類型 (Node Type), 節點與節點之間具有階層關係, 有些節點僅能包含其他節點; 有些節點僅能被包含在其他節點中; 而有些節點則兩者皆可。¹⁷

茲將 DOM 所定義的 12 種節點類型整理於表 1 中。

節點類型 (Node Type)	意義	可以擁有的子節點類型
Document	文件節點, 為 DOM 樹狀結構中最上層的節點, 代表整份 XML 文件。	Element ProcessingInstruction Comment DocumentType
DocumentFragment	文件片段節點, 由文件節點中的某段子樹所構成。	Element ProcessingInstruction

¹⁶ 同註6。

¹⁷ 同註6。

節點類型 (Node Type)	意義	可以擁有的子節點類型
		Comment Text CDATASection EntityReference
DocumentType	文件類型節點，亦即 XML 文件中由 <!DOCTYPE> 宣告的部份。	無
EntityReference	實體參照節點，亦即在 XML 文件中透過 ENTITY 型態，參照到某一實體的部份。	Element ProcessingInstruction Comment Text CDATASection EntityReference
Element	元素節點，亦即 XML 文件中的標籤部份。	Element ProcessingInstruction Comment Text CDATASection EntityReference
Attr	屬性節點，亦即 XML 文件中內含於標籤中的屬性部份。	Text EntityReference
ProcessingInstruction	處理指令節點，亦即 XML 文件的序言中由 <? ... ?> 宣告的部份，例如引用樣式表的指令 <?xml-stylesheet href="test.xml" ?>。	無
Comment	註解節點，代表 XML 文件中由 <!-- ... --> 所標注的部份。	無
Text	文字節點，例如元素內容、屬性值、實體設定值 ... 等。	無
CDATASection	CDATA 區段節點，亦即 XML 文件中由 <![CDATA[...]]> 所標注的部份。	無
Entity	實體節點，代表 XML 文件中的實體部份，包含可剖析的或不可剖析的實體。	Element ProcessingInstruction Comment Text CDATASection EntityReference
Notation	註釋節點，代表 XML 文件的 DTD 中由 <!NOTATION> 宣告的部份。	無

表1 DOM 節點類型

例如，下列是一個以 XML 撰寫之圖書目錄：¹⁸

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="show_book.xsl"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">
<!-- catalog last updated 2000-11-01-->
<catalog xmlns="http://www.example.com/catalog">
  <book id="bk101">
    <author>&#71;ambardella, Matthew</author>
    <title>XML Developer's &#x47;uide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>
      <![CDATA[An in-depth look at creating applications with XML,
        using <, >,]]> and &amp;.
    </description>
  </book>
  <book id="bk109">
    <author>Kress, Peter</author>
    <title>Paradox Lost</title>
    <genre>Science Fiction</genre>
    <price>6.95</price>
    <publish_date>2000-11-02</publish_date>
    <description>
      After an inadvertant trip through a Heisenberg Uncertainty Device,
      James Salway discovers the problems of being quantum.
    </description>
  </book>
</catalog>
```

圖2 以 XML 撰寫之圖書目錄

若將此 XML 文件以 MSXML 剖析器展開成 DOM 結構，則其最頂端的兩層節點將形成如圖 3 所示的樹狀結構。

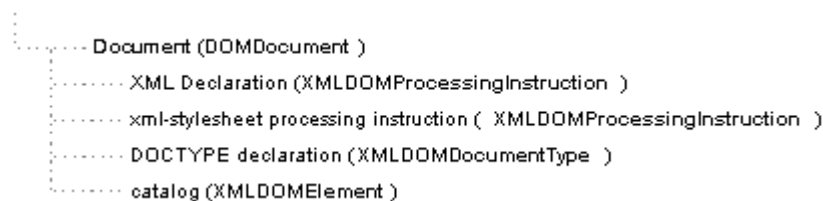


圖3 最頂端兩層節點之樹狀結構¹⁹

從圖中可以清楚的看到，最頂端的節點是 Document 節點，也就是文件本身，而

¹⁸ Microsoft, "Introduction to the DOM," available at
<<http://msdn.microsoft.com/library/psdk/xmlsdk/xmlp2lkd.htm>>

¹⁹ 同註18。

此節點之內則內含其他子節點，包括 XML 宣告節點 (XML Declaration)、樣式表處理指令節點 (Style sheet processing instruction)、文件類型宣告節點 (DOCTYPE Declaration) 和文件的根元素節點 (Root element for the document) 等。

文件的根元素節點 (在此例中為 catalog) 則又有內含了文件的實際資料內容,其結構如圖 4所示。其中包含了元素節點(Element nodes) 屬性節點(Attribute nodes) 文字節點 (Text nodes) 和 CDATA 節點 (CDATA nodes) 等。

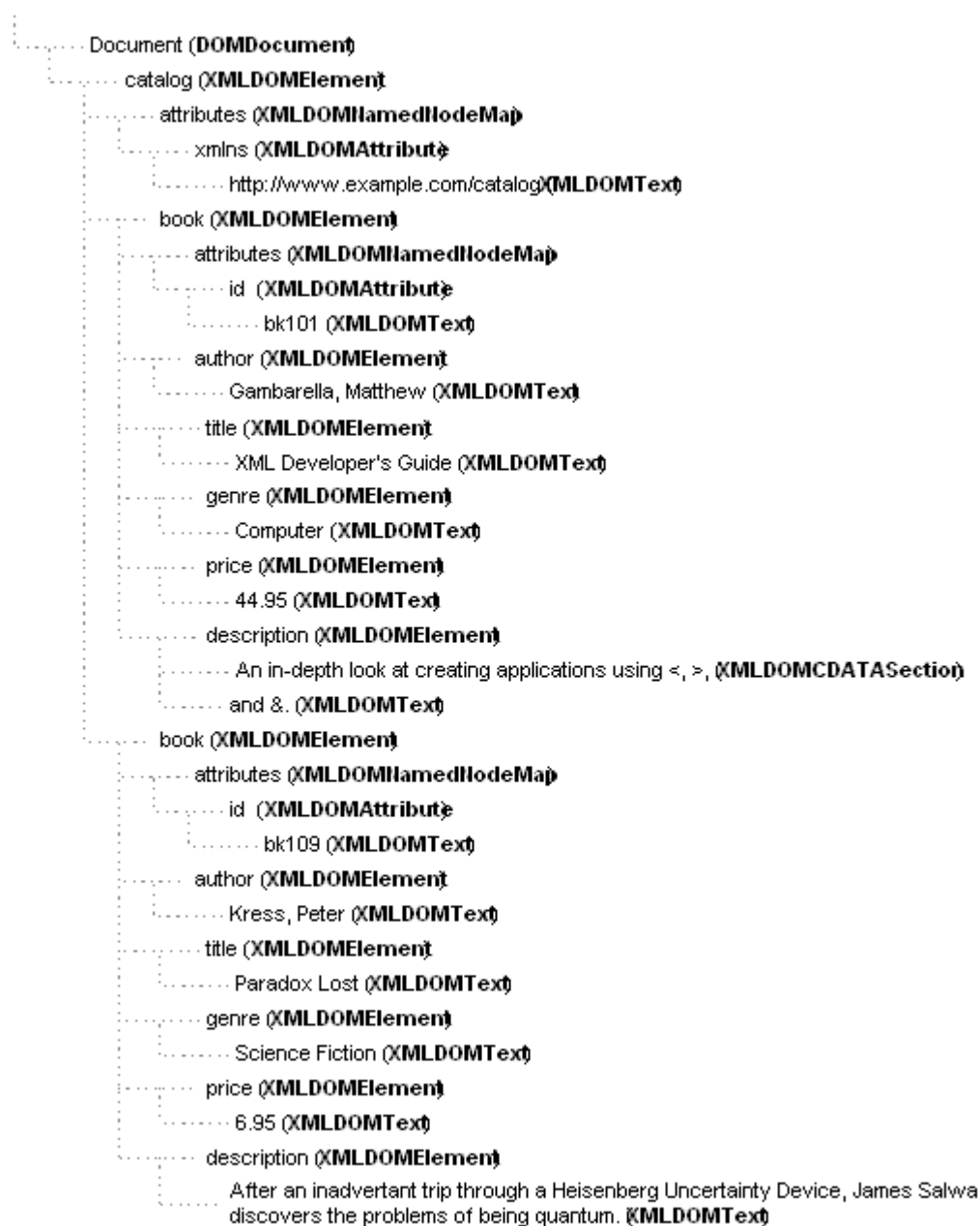


圖4 XML 文件根元素之 DOM 結構²⁰

四、DOM 核心介面

在 DOM 規範中最基本也最重要的便是其核心介面 (DOM Core Interface),

²⁰ 同註18。

是應用程式存取上述各類節點之基本管道。在 DOM 核心介面中雖然只定義了極少的物件集合和介面，卻已足夠一般軟體開發者或 Web 程式設計師使用，據以撰寫出能夠存取和操縱 HTML 或 XML 文件內容的應用程式。在 DOM 核心介面中又區分為基本介面 (Fundamental Interfaces) 與延伸介面 (Extended Interfaces) 兩大類：基本介面定義了應用程式存取 XML 或 HTML 文件皆可使用的一些通用介面，有 Document 介面 (Document Interface)、DocumentFragment 介面 (DocumentFragment Interface)、Node 介面 (Node Interface)、NodeList 介面 (NodeList Interface)、NamedNodeMap 介面 (NamedNodeMap Interface)、CharacterData 介面 (CharacterData Interface)、Attr 介面 (Attr Interface)、Element 介面 (Element Interface)、Text 介面 (Text Interface)、Comment 介面 (Comment Interface) 等；而延伸介面所定義的物件則僅供 XML 文件使用，HTML 文件則不須使用到此部份，它們是 CDATASection 介面 (CDATASection Interface)、DocumentType 介面 (DocumentType Interface)、Notation 介面 (Notation Interface)、Entity 介面 (Entity Interface)、EntityReference 介面 (EntityReference interface) 和 ProcessingInstruction 介面 (ProcessingInstruction Interface) 等。²¹此外，DOM 並引用物件導向 (Object Oriented) 概念，一一定義了這些介面的相關屬性與方法 (Method)。

(一) 基本介面

1. Document 介面

Document 介面用來代表整個 XML 或 HTML 文件，概念上，它是整棵樹的樹根，為 DOM 樹狀結構的最上層節點，提供存取資料的主要管道。其屬性及方法分別歸納如表 2 和表 3 所示。

屬性名稱	含意
docType	代表此份 XML 文件中的文件類型宣告部份。
DocumentElement	代表此份 XML 文件的根元素。

表2 Document 介面之屬性及用途

方法名稱	用途	參數：意義	回傳值
createElement	在此文件中建立一個元素節點。	tagName：元素之標籤名稱。	Element 物件。
createDocumentFragment	在此文件中建立一個文件片段。	無。	DocumentFragment 物件。
createTextNode	建立一個文字節點。	data：節點內容。	Text 物件。
createComment	建立一個註解節點。	data：節點內容。	Comment 物件。

²¹ Mike Champion, ArborText, Steve Byrne, JavaSoft, Gavin Nicol, Inso EPS, Lauren Wood, SoftQuad, Inc., "Document Object Model (Core) Level 1," 1998-10-01, available at <<http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html>>.

方法名稱	用途	參數：意義	回傳值
createCDATASection	建立一個 CDATA 區段節點。	data：節點內容。	CDATASection 物件。
createProcessingInstruction	建立一個指令處理節點。	target：處理指令的目標部份。 data：節點內容。	ProcessingInstruction 物件。
createAttribute	建立一個 Attr 節點。	name：屬性名稱。	Attr 物件。
createEntityReference	建立一個實體參照節點。	name：被參照之實體名稱。	EntityReference 物件。
getElementByTagName	取得符合由指定標籤名稱所構成的節點群。	tagName：標籤名稱。	NodeList 物件。

表3 Document 介面之方法及用途

2. Node 介面

Node 介面代表 DOM 樹狀結構中任一個特定的節點。由於在 DOM 模型中共有十二種不同類型的節點，因此 Node 介面便定義十二個常數符號來代表這十二種不同類型的節點，如表 4 所示。

常數符號	意義
ELEMENT_NODE	表示此節點為一個元素節點
ATTRIBUTE_NODE	表示此節點為一個屬性節點
TEXT_NODE	表示此節點為一個文字節點
CDATA_SECTION_NODE	表示此節點為一個 CDATA 區段
ENTITY_REFERENCE_NODE	表示此節點為一個實體參照節點
ENTITY_NODE	表示此節點為一個實體節點
PROCESSING_INSTRUCTION_NODE	表示此節點為一個處理指令節點
COMMENT_NODE	表示此節點為一個註解節點
DOCUMENT_NODE	表示此節點為一個文件節點
DOCUMENT_TYPE_NODE	表示此節點為一個文件類型節點
DOCUMENT_FRAGMENT_NODE	表示此節點為一個文件片段節點
NOTATION_NODE	表示此節點為一個註釋節點

表4 常數符號與節點類型對照表

Node 介面同時定義了許多屬性與方法作為存取資料的基本介面，分別歸納如表 5 和表 6 所示。

屬性名稱	含意
nodeName	目前節點的名稱
nodeValue	目前節點的內容
nodeType	目前節點的類型

屬性名稱	含意
parentNode	目前節點的父節點（上層節點）
childNodes	由目前節點的子節點所構成的節點群
firstChild	目前節點的第一個子節點（下層節點）
lastChild	目前節點的最後一個子節點（下層節點）
previousSibling	目前節點的前一個兄弟節點（同層節點）
nextSibling	目前節點的後一個兄弟節點（同層節點）
attributes	由目前節點的屬性所構成的群體
ownerDocument	目前節點所屬的文件物件

表5 Node 介面之屬性及用途

方法名稱	用途	參數：意義	回傳值
insertBefore	在目前的節點之下，新增一個新子節點到參考子節點之前。若參考子節點不存在，則新子節點被加到子節點群的最後一個。	newChild：新子節點名稱。 refChild：參考子節點名稱。	被新增的節點物件。
replaceChild	在目前的節點之下，以一個新子節點取代舊子節點。	newChild：新子節點。 oldChild：即將被取代的舊子節點。	被取代的節點物件
removeChild	在目前的節點之下，移除一個現有的子節點。	oldChild：即將被取代的子節點。	被移除的節點物件。
appendChild	在目前的節點之下，新增一個新子節點到子節點群的最後一個。	newChild：新子節點。	被新增的節點物件。
hasChildNodes	偵測目前節點是否擁有子節點。	無。	True：有子節點。 False：無子節點。
cloneNode	複製目前節點。	deep：若為 True，則遞迴式（Recursive）的複製此節點之所有子樹；否則只複製此節點本身。	被複製的節點。

表6 Node 介面之方法及用途

3. NodeList 介面

NodeList 介面代表 DOM 樹狀結構中由若干節點所構成的節點群，其中的每個成員皆為一個 Node 物件。NodeList 介面所定義的屬性及方法分別歸納如表 7 和表 8 所示：

屬性名稱	含意
length	整個 NodeList 節點群中的節點個數，每個節點有一個節點編號，

屬性名稱	含意
	範圍由 0 ~ length-1。

表7 NodeList 介面之屬性及用途

法名稱	用途	參數：意義	回傳值
item	傳回 NodeList 節點群中某個指定的節點。	index: 節點群中的節點編號。	被指定的第 index 個節點，若 index 比最大的節點編號大則傳回 null。

表8 NodeList 介面之方法及用途

4. NamedNodeMap 介面

NamedNodeMap 介面代表可以藉由名稱存取的節點所成的集合，但 NamedNodeMap 並非從 NodeList 繼承而來，此點須特別注意。NamedNodeMap 介面所定義的屬性及方法分別歸納如表 9 和表 10 所示：

屬性名稱	含意
length	整個 NamedNodeMap 節點群中的節點個數。每個節點有一個節點編號，範圍由 0 ~ length-1。

表9 NamedNodeMap 介面之屬性及用途

方法名稱	用途	參數：意義	回傳值
getNamedItem	傳回一個指名的節點	name：指定的節點名稱	指名的節點，若無則傳回 null
setNamedItem	加入一個指名的節點。	arg：將被加入的節點，若已有一個同名的節點存在則取代之。	若新節點取代了舊節點，則傳回舊節點，否則傳回 null。
removeNamedItem	移除一個指名的節點。	name：欲移除的節點名稱。	若指名的節點存在則傳回被移除的節點，若不存在則傳回 null。
item	傳回 NamedNodeMap 節點群中某個指定的節點	index：節點群中的節點編號。	被指定的第 index 個節點，若 index 比最大的節點編號大則傳回 null。

表10 NamedNodeMap 介面之方法及用途

5. CharacterData 介面

CharacterData 介面為 Node 介面之延伸，以提供更簡易的管道存取字元資料。茲將 CharacterData 介面所定義的屬性及方法歸納如表 11 和表 12 所示：

屬性名稱	含意
data	此節點之字元資料。
length	字元資料的長度。

表11 CharacterData 介面之屬性及用途

方法名稱	用途	參數：意義	回傳值
substringData	從節點中擷取一段子字串資料。	offset: 欲擷取之子字串的起點。 count: 欲擷取之子字串的字元數。	擷取到的子字串。 若 offset 與 count 之和超過 length, 則傳回起點起所有字元。
appendData	將一字串接於節點資料之後。	arg: 欲被加入的字串。	無。
insertData	將一字串插入指定的節點資料位置中。	offset: 欲插入字串的起點。 arg: 欲被插入的字串。	無。
deleteData	從節點中刪除一段子字串資料。	offset: 欲刪除之子字串的起點。 count: 欲刪除之子字串的字元數。 若 offset 與 count 之和超過 length, 則刪除起點起所有字元。	無。
replaceData	將節點中一段子字串資料以另一子字串取代之。	offset: 欲被取代之子字串的起點。 count: 欲被取代之子字串的字元數。 arg: 取代的字串。 若 offset 與 count 之和超過 length, 則取代起點起所有字元。	無。

表12 CharacterData 介面之方法及用途

6. Attr 介面

Attr 介面代表某個特定的 Element 物件中的一個屬性。由於 Attr 介面繼承自 Node 介面，因此擁有 Node 的屬性與方法。而 Attr 介面自身另外定義的屬性則歸納如表 13 所示：

屬性名稱	含意
name	傳回此 Attr 物件之名稱。

屬性名稱	含意
specified	若此 Attr 物件明訂須給值，則此屬性為 True，否則為 False。
value	傳回此 Attr 物件之值。

表13 Attr 介面之屬性及用途

7. Element 介面

Element 介面代表 XML 文件中某個特定的元素，對於 XML 文件的作者而言便是所謂的標籤。由於 Element 介面亦繼承自 Node 介面，因此擁有 Node 的屬性與方法。而 Element 介面自身另外定義的屬性及方法則歸納如表 14和表 15所示：

性名稱	含意
tagName	標籤名稱。

表14 Element 介面之屬性及用途

方法名稱	用途	參數：意義	回傳值
getAttribute	藉由指定標籤中的屬性名稱而取得其屬性值。	name：屬性名稱。	屬性值字串。
setAttribute	在標籤中加入一個新屬性，若該屬性已經存在，則以新設定值取代原設定值。	name：屬性名稱。 value：屬性值字串。	無。
removeAttribute	藉由指定標籤中的屬性名稱而刪除該屬性。	name：屬性名稱。	無。
getAttributeNode	藉由指定標籤中的屬性名稱而取得代表該屬性的 Attr 物件。	name：屬性名稱。	Attr 物件。
setAttributeNode	在標籤中加入一個新的屬性節點，若該節點已經存在，則以新節點取代原節點。	newAttr：欲新增的 Attr 物件。	若已存在與新節點同名的舊節點，則傳回被取代的舊節點，否則傳回 null。
removeAttributeNode	刪除指定的屬性節點。	oldAttr：欲刪除的 Attr 物件。	被刪除的 Attr 物件。
getElementsByTagName	取得目前元素所屬子節點中符合指定標籤名稱之所有元素所構成的 NodeList 物件。	name：欲擷取的節點標籤名稱。	符合 name 之所有元素所構成的 NodeList 物件。

方法名稱	用途	參數：意義	回傳值
normalize	將目前元素所屬之 Text 節點合併成正規化格式。	無。	無。

表15 Element 介面之方法及用途

8. Text 介面

Text 介面代表元素或屬性中之文字資料內容。茲將 Text 介面所定義的方法歸納如表 16 所示：

方法名稱	用途	參數：意義	回傳值
splitText	將此 Text 節點一分為二。	offset：字串切割處，由 0 算起。	新的 Text 節點。

表16 Text 介面之方法及用途

9. Comment 介面

Comment 介面代表文件中的註解內容，亦即介於 <!-- 與 --> 之間的所有字元所構成之字串。Comment 介面並未定義任何屬性與方法。

(二) 延伸介面

1. CDATASection 介面

CDATASection 介面代表 XML 文件中的 CDATA 區段。CDATA 區段的主要用途是用來標注一段內含特殊符號或標籤符號之文字，此段文字不會被剖析器所剖析。CDATA 區段以 <![CDATA[作為啟始標籤，以]]> 作為結束標籤。CDATASection 介面繼承了 CharacterData 介面和 Text 介面的屬性與方法。

2. DocumentType 介面

DocumentType 介面代表 XML 文件中文件類型宣告 (DTD) 的部份，亦即 DOCTYPE 的內容。茲將 DocumentType 介面所定義的屬性歸納如表 17 所示：

屬性名稱	含意
name	文件類型宣告中所引用的 DTD 的名稱。
entities	由 DTD 宣告中所有實體部份所組成的 NamedNodeMap 物件。
notations	由 DTD 宣告中所有註釋部份所組成的 NamedNodeMap 物件。

表17 DocumentType 介面之屬性及用途

3. Notation 介面

Notation 介面代表 DTD 中所宣告的註釋部份。茲將 Notation 介面所定義的屬性歸納如表 18 所示：

屬性名稱	含意
publicID	此註釋的公用識別名稱。
systemID	此註釋的系統識別名稱。

表18 Notation 介面之屬性及用途

4. Entity 介面

Entity 介面代表 XML 文件中的實體部份，而非 DTD 中的實體宣告，此點應特別留意。茲將 Entity 介面所定義的屬性歸納如表 19 所示：

屬性名稱	含意
publicID	此實體的公用識別名稱。
systemID	此實體的系統識別名稱。
notationName	對於不可剖析的實體，此屬性值為其對應的註釋名稱；對於可剖析的實體，此屬性值為 null。

表19 Entity 介面之屬性及用途

5. EntityReference 介面

EntityReference 介面代表 XML 文件中的實體參照部份，此介面繼承了 Node 介面的屬性與方法。

6. ProcessingInstruction 介面

ProcessingInstruction 介面代表 XML 文件中的處理指令部份，亦即由啟始標籤 `<?` 和結束標籤 `?>` 所標注的部份。茲將 ProcessingInstruction 介面所定義的屬性歸納如表 20 所示：

屬性名稱	含意
target	代表此處理指令的目標，亦即處理指令中的第一個標記 (Token)。
data	代表此處理指令的內容，亦即從 target 之後的第一個非空白字元，到結束標籤 <code>?></code> 之前的所有字元所組成的字串。

表20 ProcessingInstruction 介面之屬性及用途

五、DOM 之實作

一、MSXML 剖析器

W3C 的 DOM 規格書僅是一份規範應用程式與文件之間的介面標準，至於要使用何種程式語言或作業平台加以實現，則未在規範之列，可由各家軟體廠商或開發人員自行決定。目前，已有許多符合 DOM API 規範的剖析器 (Parser) 被開發出來，如：

- ◆ Microsoft 的 MSXML²²
- ◆ IBM 的 XML4C²³和 XML4J²⁴
- ◆ Oracle 的 XML parser for C²⁵和 XML parser for Java²⁶
- ◆ Sun 的 Java Project X²⁷
- ◆ Daniel Veillard 的 GNOME XML Library²⁸
- ◆ 其他

以下以微軟公司所開發的 MSXML 為例，說明 DOM 在文件剖析與處理上所扮演的重要角色。微軟自從 1998 年起，便開始全力投入 XML 技術支援與開發，其瀏覽器 Internet Explorer 5.0 更是市面上最早支援 XML 規格的瀏覽器之一。²⁹要取得最新版本的 MSXML 剖析器可以至微軟 MSDN 網站下載，MSXML 不僅支援 DOM 物件模型，更進一步滿足 XSLT 和 XPath 規範，使得程式人員可以更得心應手的開發 XML 處理程式。³⁰

表 21 乃依據 MSXML 中所支援的 DOM 介面物件，歸納出較重要且經常使用者，本節並以三種不同的程式語言為例(Visual Basic、VBScript 和 JavaScript)，闡釋其語法及用法，以作為實作 XML 文件處理程式的依據。

DOM 物件	說明
DOMDocument	代表樹狀結構中的最上層節點。此物件為 DOM 中的 Document 介面實作。
IXMLDOMNode	代表樹狀結中任一個特定的節點，為 XML 物件模型中存取資料的基本介面，其 nodeType 屬性可以用來判斷節點類型。此物件為 DOM 中的 Node 介面實作。
IXMLDOMNodeList	由 IXMLDOMNode 節點物件所組成的節點陣列。此物件為 DOM 中的 NodeList 介面實作。
IXMLDOMParseError	此物件回傳處理過程中的錯誤訊息，包括錯誤代碼、錯誤行號、錯誤位置和錯誤說明。

表21 MSXML 中較常用的 DOM 介面物件

二、DOMDocument 物件

DOMDocument 物件用來代表整份 XML 或 HTML 文件，概念上，它是整棵樹的樹根，為 DOM 樹狀結構的最上層節點，提供存取資料的主要管道。

(一) 建立 DOMDocument 物件

要建立一個 DOMDocument 物件，可使用如下語法：

²² Microsoft, "MSDN Online Download: MSXML Parser 3.0 Release," Last updated: 10/27/2000, <<http://msdn.microsoft.com/downloads/webtechnology/xml/msxml.asp>>.

²³ IBM alphaWorks, "XML parser for C++," <<http://www.alphaworks.ibm.com/tech/xml4c>>.

²⁴ IBM alphaWorks, "XML parser for Java," <<http://alphaworks.ibm.com/tech/xml4j>>.

²⁵ Oracle, "XML: Parser for C v2," <http://technet.oracle.com/tech/xml/parser_c2/>.

²⁶ Oracle, "XML: Parser for Java v2," <http://technet.oracle.com/tech/xml/parser_java2/>.

²⁷ Sun, "Java Project X", <<http://developer.java.sun.com/developer/earlyAccess/xml/index.html>>.

²⁸ <http://xmlsoft.org/>

²⁹ Internet Explorer 至 2000 年 12 月止最新版本為 5.5，中文版可至以下 URL 下載 <<http://www.microsoft.com/taiwan/products/ie/default.htm>>.

³⁰ Microsoft Cooperation, "Microsoft XML 3.0 - XML Developer's Guide: Introduction to the DOM," <<http://msdn.microsoft.com/library/psdk/xmlsdk/xmlp2lkd.htm>>

- ◆ Visual Basic 語法：

```
Dim xmlDoc As New Msxml.DOMDocument
```

- ◆ VBScript 語法：

```
Dim xmlDoc  
xmlDoc = CreateObject("Msxml.DOMDocument")
```

- ◆ JavaScript 語法：

```
var xmlDoc = new ActiveXObject("Msxml.DOMDocument");
```

- ◆ 參數說明：

參數	說明
<i>xmlDoc</i>	物件變數名稱
Msxml.DOMDocument	物件類別 (Class)

(二) 載入 XML 文件

DOMDocument 物件提供了一個 load 方法 (Method)，作為載入文件之用，其語法如下：

- ◆ Visual Basic 語法：

```
boolValue = xmlDoc.load(xmlSource)
```

- ◆ VBScript 語法：

```
boolValue = xmlDoc.load(xmlSource)
```

- ◆ JavaScript 語法：

```
boolValue = xmlDoc.load(xmlSource);
```

- ◆ 參數說明：

參數	說明
<i>xmlDoc</i>	DOMDocument 物件
<i>xmlSource</i>	一個字串，其內容為即將被載入的 XML 檔案所在之 URL。
<i>BoolValue</i>	回傳值，為一布林值：True 表示載入成功，False 表示載入失敗。

(三) 儲存 XML 文件

DOMDocument 物件提供了一個 save 方法 (Method)，作為儲存文件之用，

其語法如下：

- ◆ Visual Basic 語法：

xmlDoc.save(destination)

- ◆ VBScript 語法：

xmlDoc.save(destination)

- ◆ JavaScript 語法：

xmlDoc.save(destination);

- ◆ 參數說明：

參數	說明
<i>xmlDoc</i>	DOMDocument 物件
<i>destination</i>	存檔路徑與檔名

三、IXMLDOMNode 物件

IXMLDOMNode 物件代表 DOM 樹狀結構中任一個特定的節點，IXMLDOMNode 提供許多屬性與方法作為 XML 物件模型中存取資料的基本介面。茲闡釋其較重要的屬性與方法如下：

(一) 獲取節點相關資訊

- nodeType 屬性：

這是一個唯讀的屬性，屬性值代表該節點的類型，其含意如表 22 所示：

數值	常數符號	意義
1	NODE_ELEMENT	表示此節點為一個元素節點。此類節點可有以下類型之子節點：元素、文字、註解、處理指令、CDATA 區段和實體參照等；元素節點亦可以是文件、文件片段、實體參照和其他元素的子節點。
2	NODE_ATTRIBUTE	表示此節點為一個屬性節點。此類節點可有以下類型之子節點：文字和實體參照。
3	NODE_TEXT	表示此節點為一個文字節點。此類節點不能有任何子節點，卻可以是屬性、文件片段、元素和實體參照的子節點。
4	NODE_CDATA_SECTION	表示此節點為一個 CDATA 區段。此類節點不能有任何子節點，卻可以是文件片段、元素和

數值	常數符號	意義
		實體參照的子節點。
5	NODE_ENTITY_REFERENCE	表示此節點為一個實體參照節點。此類節點可以擁有以下類型的子節點：元素、處理指令、註解、文字、CDATA 區段、和實體參照等；也可以是屬性、文件片段、元素和實體參照的子節點。
6	NODE_ENTITY	表示此節點為一個實體節點。此類節點可以擁有文字、實體參照等子節點；亦可以是文件類型的子節點。
7	NODE_PROCESSING_INSTRUCTION	表示此節點為一個處理指令節點。此類節點不能有任何子節點，卻可以是文件、文件片段、元素和實體參照之子節點。
8	NODE_COMMENT	表示此節點為一個註解節點。此類節點不能有任何子節點，卻可以是文件、文件片段、元素和實體參照之子節點。
9	NODE_DOCUMENT	表示此節點為一個文件節點為 DOM 結構樹的最頂層。此類節點可以擁有元素、處理指令、註解和文件類型等子節點，但其本身不可以是任何節點的子節點。
10	NODE_DOCUMENT_TYPE	表示此節點為一個以 <!DOCTYPE> 宣告的文件類型節點。此類節點可以擁有下列類型的子節點：註釋、實體；也可以是文件節點的子節點。
11	NODE_DOCUMENT_FRAGMENT	表示此節點為一個文件片段節點。此類節點可以擁有元素、處理指令、註解、文字、CDATA 區段和實體參照等子節點，但其本身不可以是任何節點的子節點。
12	NODE_NOTATION	表示此節點為一個註釋節點。此類節點不允許擁有任何子節點，但可以是文件類型的子節點。

表22 nodeType 屬性之含意

- nodeTypeString 屬性：
這也是一個唯讀的屬性，其屬性值為 nodeType 所對應的字串。nodeType 與 nodeTypeString 對照如表 23 所示：

nodeType 屬性值	對應的 nodeTypeString 屬性值
NODE_ATTRIBUTE	attribute
NODE_CDATA_SECTION	cdatasection
NODE_COMMENT	comment
NODE_DOCUMENT	document
NODE_DOCUMENT_FRAGMENT	documentfragment
NODE_DOCUMENT_TYPE	documenttype
NODE_ELEMENT	element
NODE_ENTITY	entity
NODE_ENTITY_REFERENCE	entityreference
NODE_NOTATION	notation
NODE_PROCESSING_INSTRUCTION	processinginstruction
NODE_TEXT	text

表23 nodeType 與 nodeTypeString 對照表

- nodeName 屬性：
此屬性為節點名稱，也是一個唯讀的屬性，其屬性值也是依據 nodeType 的不同而不同。nodeType 與 nodeName 的對照如表 24 所示。

nodeType 屬性值	對應的 nodeName 屬性值
NODE_ATTRIBUTE	傳回屬性名稱
NODE_CDATA_SECTION	傳回字串 "#cdata-section"
NODE_COMMENT	傳回字串 "#comment".
NODE_DOCUMENT	傳回字串 "#document".
NODE_DOCUMENT_TYPE	傳回文件類型名稱，亦即<!DOCTYPE xxx ...> 中的 xxx
NODE_DOCUMENT_FRAGMENT	傳回字串 "#document-fragment"
NODE_ELEMENT	傳回 XML 標籤名稱，若有 namespace 則連同前導符號一起傳回
NODE_ENTITY	傳回實體名稱
NODE_ENTITY_REFERENCE	傳回實體參照名稱，不含 "&" 和 ";"
NODE_NOTATION	傳回註釋名稱
NODE_PROCESSING_INSTRUCTION	傳回在處理指令 <? 後的第一個字符
NODE_TEXT	傳回字串 "#text"

表24 nodeType 與 nodeName 對照表

- hasChildNodes 方法：
此方法可用來偵測目前節點是否具有子節點；其傳回值若是 True，表示此節點有子節點；若是 False 則表示此節點沒有子節點。
- xml 屬性：
此屬性包含此節點及其下屬節點的 XML 內容，亦為一唯讀屬性。

(二) 讀取或設定節點資料

- nodeValue 屬性：
此屬性存放節點的資料內容，可讀寫。

- text 屬性：
此屬性存放節點的資料內容，可讀寫。

(三) 在節點間穿梭

使用下列屬性可以直接在子節點間切換穿梭。

- childNodes 屬性：
此節點的下一層子節點，為一個節點陣列。
- parentNode 屬性：
此節點的上一層父節點。
- firstChild 屬性：
此節點的第一個子節點。
- lastChild 屬性：
此節點的最後一個子節點。
- attributes 屬性：
若此節點為一個元素節點，則此屬性為其屬性所構成之陣列。

(四) 子節點之處理

對於某個特定的節點，可以使用以下方法對其本身及其子節點進行新增、刪除、取代、插入等操作：

- insertBefore 方法：
插入一個新節點到參考節點之前。其語法如下，其中 *oXMLDOMNode* 指的是被操作的原始 Node 物件，而 *objXMLDOMNode* 指的是操作後回傳的 Node 物件：

- ◆ Visual Basic 語法：

```
Set objXMLDOMNode = oXMLDOMNode.insertBefore(newChild, refChild)
```

- ◆ VBScript 語法：

```
Set objXMLDOMNode = oXMLDOMNode.insertBefore(newChild, refChild)
```

- ◆ JavaScript 語法：

```
objXMLDOMNode = oXMLDOMNode.insertBefore(newChild, refChild);
```

- ◆ 參數說明：

參數	說明
<i>oXMLDOMNode</i>	原始的 Node 物件，亦即欲新增子節點的節點。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即新增的子節點。
<i>newChild</i>	新增子節點。
<i>refChild</i>	參考子節點。

- replaceChild 方法：
將某特定子節點以新節點取代。可使用如下語法：
- ◆ Visual Basic 語法：

Set *objXMLDOMNode* = *oXMLDOMNode*.replaceChild(*newChild*, *oldChild*)

◆ VBScript 語法：

Set *objXMLDOMNode* = *oXMLDOMNode*.replaceChild(*newChild*, *oldChild*)

◆ JavaScript 語法：

objXMLDOMNode = *oXMLDOMNode*.replaceChild(*newChild*, *oldChild*);

◆ 參數說明：

參數	說明
<i>oXMLDOMNode</i>	原始的 Node 物件，亦即欲進行子節點取代操作的節點。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即新增的子節點。
<i>newChild</i>	新子節點。
<i>oldChild</i>	將被取代的子節點。

● removeChild 方法：

刪除此節點的某個特定的子節點。可使用如下語法：

◆ Visual Basic 語法：

Set *objXMLDOMNode* = *oXMLDOMNode*.removeChild(*childNode*)

◆ VBScript 語法：

Set *objXMLDOMNode* = *oXMLDOMNode*.removeChild(*childNode*)

◆ JavaScript 語法：

objXMLDOMNode = *oXMLDOMNode*.removeChild(*childNode*);

◆ 參數說明：

參數	說明
<i>oXMLDOMNode</i>	欲進行子節點刪除操作的原始 Node 物件。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即被刪除的子節點。
<i>childChild</i>	欲被刪除的子節點。

● appendChild 方法：

添加一個新的子節點到此節點的子節點陣列的最後面。可使用如下語法：

◆ Visual Basic 語法：

Set *objXMLDOMNode* = *oXMLDOMNode*.appendChild(*newChild*)

◆ VBScript 語法：

```
Set objXMLDOMNode = oXMLDOMNode.appendChild(newChild)
```

◆ JavaScript 語法：

```
objXMLDOMNode = oXMLDOMNode.appendChild(newChild);
```

◆ 參數說明：

參數	說明
<i>oXMLDOMNode</i>	欲進行子節點添加操作的原始 Node 物件。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即被添加的子節點。
<i>newChild</i>	欲被添加的子節點。

● cloneNode 方法：

複製此節點。可使用如下語法：

◆ Visual Basic 語法：

```
Set objXMLDOMNode = oXMLDOMNode.cloneNode(deep)
```

◆ VBScript 語法：

```
Set objXMLDOMNode = oXMLDOMNode.cloneNode(deep)
```

◆ JavaScript 語法：

```
objXMLDOMNode = oXMLDOMNode.cloneNode(deep);
```

◆ 參數說明：

參數	說明
<i>oXMLDOMNode</i>	欲進行子節點複製操作的原始 Node 物件。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即被複製的節點。
<i>deep</i>	布林值，若為 True，則遞迴式的複製此節點之所有子樹；否則只複製此節點本身。

四、IXMLDOMNodeList 物件

IXMLDOMNodeList 物件代表 DOM 樹狀結構中由若干節點所構成的節點陣列，其中的每個成員皆為一個 IXMLDOMNode 物件。茲闡釋 IXMLDOMNodeList 物件之重要屬性及方法如下：

● length 屬性：

代表整個節點陣列的長度，亦即成員個數。

● item 方法：

傳回某個特定的節點。可使用如下語法：

◆ Visual Basic 語法：

Set *objXMLDOMNode* = *oXMLDOMNodeList*.item(*index*)

◆ VBScript 語法：

Set *objXMLDOMNode* = *oXMLDOMNodeList*.item(*index*)

◆ JavaScript 語法：

objXMLDOMNode = *oXMLDOMNodeList*.item(*index*);

◆ 參數說明：

參數	說明
<i>oXMLDOMNodeList</i>	欲進行節點巡航操作的原始 NodeList 物件。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即節點群中由 <i>index</i> 所指定的節點。
<i>index</i>	長整數，節點編號，由 0 開始。

● nextNode 方法：

傳回目前節點的下一個節點。可使用如下語法：

◆ Visual Basic 語法：

Set *objXMLDOMNode* = *oXMLDOMNodeList*.nextNode

◆ VBScript 語法：

Set *objXMLDOMNode* = *oXMLDOMNodeList*.nextNode

◆ JavaScript 語法：

objXMLDOMNode = *oXMLDOMNodeList*.nextNode();

◆ 參數說明：

參數	說明
<i>oXMLDOMNodeList</i>	欲進行節點巡航操作的原始 NodeList 物件。
<i>objXMLDOMNode</i>	傳回的 Node 物件，亦即節點群中的下一個節點。

五、IXMLDOMParseError 物件

剖析過程中若有錯誤發生，可由此一物件得知相關的錯誤訊息。茲闡釋 IXMLDOMParseError 物件之重要屬性及方法如下：

- errorCode 屬性：
此為一個唯讀屬性，代表剖析錯誤代碼。
- filepos 屬性：

- 此為一個唯讀屬性，代表錯誤發生處之絕對位置。
- line 屬性：
此為一個唯讀屬性，代表錯誤發生處之行號。
- linepos 屬性：
此為一個唯讀屬性，代表錯誤發生處位於該行之字元位置。
- reason 屬性：
此為一個唯讀屬性，代表錯誤發生之原因。
- srcText 屬性：
此為一個唯讀屬性，傳回發生錯誤之行的全文。
- url 屬性：
此為一個唯讀屬性，傳回發生錯誤之 XML 文件的 URL。

六、DOM 實驗系統：XML 線上剖析器實作

本節嘗試以上述的 MSXML 為核心，設計一個 XML 線上剖析器實驗系統 (Prototype)，XML 文件作者只要利用一般瀏覽器即可連線至本系統，即可在線上進行 XML 文件之剖析，剖析結果將轉換成標準網頁模式輸出，不論使用者之瀏覽器是否有支援 XML 功能皆能相容。

此系統前端提供一個輸入介面，供使用者輸入一份 XML 文件，後端則利用 ASP 程式進行剖析，將使用者所傳送的 XML 文件內容依 DOM 樹狀結構展開，然後以樹狀階層式結構輸出。假設現有一份 XML 文件內容如下所述：

```
<?xml version="1.0" encoding="BIG5" ?>
<!DOCTYPE 書籍目錄 [
<!ELEMENT 書籍目錄 (書籍*)>
<!ELEMENT 書籍 (ISBN, 書名, 作者+, 出版社, 定價)>
  <!ATTLIST 書籍
    序號 CDATA #REQUIRED>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT 書名 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
<!ELEMENT 出版社 (#PCDATA)>
<!ELEMENT 定價 (#PCDATA)>
]>
<書籍目錄>
  <書籍 序號="1">
    <ISBN>957-23-0721-5</ISBN>
    <書名>精通 Visual Basic 6 程式設計</書名>
    <作者>林信成</作者>
    <出版社>第三波</出版社>
    <定價>490</定價>
  </書籍>
  <書籍 序號="2">
    <ISBN>957-22-2504-9</ISBN>
    <書名>多媒體網路：趨勢、技術、應用</書名>
    <作者>林盈達</作者>
```

<出版社>松崗</出版社>
<定價>300</定價>
</書籍>
</書籍目錄>

本系統所提供之輸入畫面如圖 5 所示。

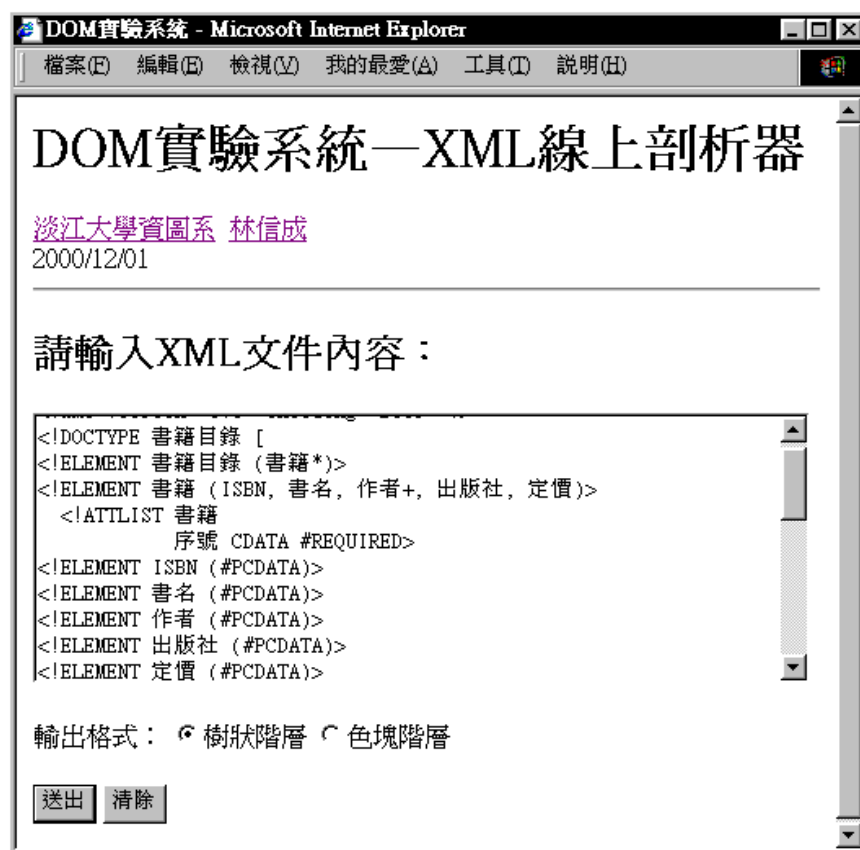


圖5 XML 線上剖析器輸入介面

本系統目前共提供兩種輸出格式供使用者選擇，分別是「樹狀階層」和「色塊階層」。其中，「樹狀階層」格式是將 XML 文件剖析結果依 DOM 所定義之樹狀結構展開，並將各節點之類型、名稱、內容依序列出成階層式架構，其輸出畫面如圖 6 所示；而「色塊階層」輸出畫面則如圖 7 所示，除同樣具備階層式輸出格式外，另將各節點之類型、名稱、內容分別以不同顏色之色塊標出，以便識別，並將各不同階層以分欄方式編排，使得同一階層能夠對齊，因此其版面格式遠較「樹狀階層」醒目，至於是否較為整齊、美觀則見仁見智。



圖6 XML 線上剖析器之「樹狀階層」輸出畫面



圖7 XML 線上剖析器之「色塊階層」輸出畫面

此外，本系統乃為一具備有效性檢驗之剖析器（Validation Parser），因此若是 XML 文件未依據 DTD 之定義撰寫，則會產生剖析錯誤。例如，在上例中的第一本書，若是漏了「書籍」元素的序號屬性，如下所示：

```

<書籍>
  <ISBN>957-23-0721-5</ISBN>
  <書名>精通 Visual Basic 6 程式設計</書名>
  <作者>林信成</作者>
  <出版社>第三波</出版社>
  <定價>490</定價>
</書籍>

```

由於此屬性在 DTD 中以 #REQUIRED 宣告為必要，所以此 XML 文件便不滿足有效性，本系統在剖析時將輸出如下錯誤訊息：

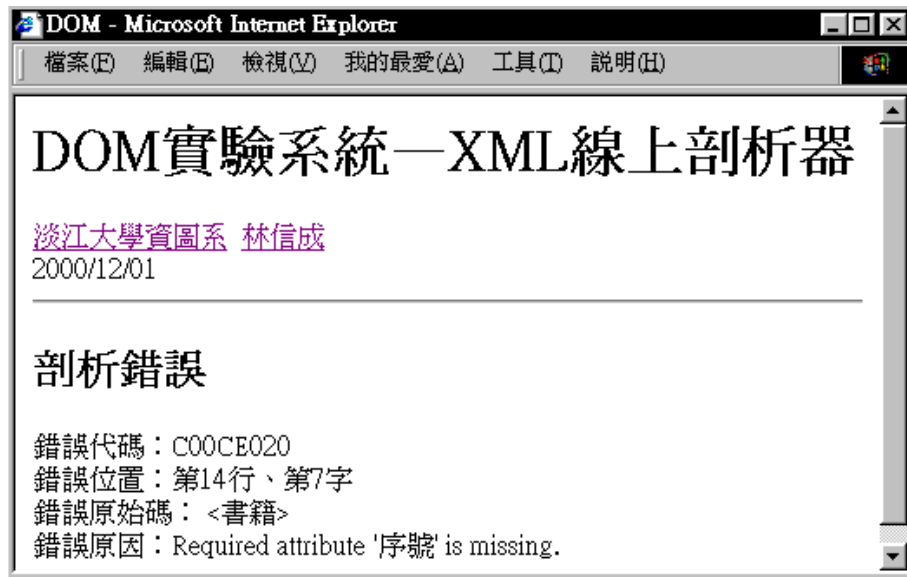


圖8 剖析錯誤訊息之輸出畫面

七、結論

本文基於 W3C 所制訂的文件物件模型，深入瞭解其定義、模型、結構、介面規範，並藉由系統實作、程式撰寫等方式，解析 DOM 之運作原理，深入瞭解 XML 文件的邏輯結構以及存取、處理、操縱的方法，並設計一個 XML 線上剖析器，以供 XML 文件作者利用一般瀏覽器在線上進行 XML 文件之剖析，並將剖析結果轉換成標準網頁模式輸出，不論使用者之瀏覽器是否有支援 XML 功能皆能相容。