國立交通大學

資訊工程學系

碩士論文

基於 BODE 擷取系統的自動化網頁再生技術

Automatic Web Page Regeneration Technique Based on BODE Extraction System

研究生：周自強

指導教授：吳毅成

陳隆彬

中華民國九十四年七月

基於 BODE 擷取系統的自動化網頁再生技術

Automatic Web Page Regeneration Technique Based on BODE Extraction System

研 究 生：周自強　　Student：Tzu-Chiang Chou

指導教授：吳毅成博士　　Advisor：Dr. I-Chen Wu

陳隆彬博士　　Dr. Lung-Pin Chen

國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science

National Chiao Tung University

in

partial Fulfillment of the Requirements

for the Degree of Master

in

Computer Science and Information Engineering

July 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年七月

基於 BODE 擷取系統的自動化網頁再生技術

研究生：周自強　　指導教授：吳毅成
陳隆彬

國立交通大學資訊工程學系 碩士班

摘　要

為了整合各個應用領域的多樣化的網路電子文件，在擴充性、網路傳輸、資訊交換、及資料整合管理等各方面都具有優越特性之 XML 標準應運而生。目前 XML 文件的轉換大多以 XSLT 語法來表示，例如，XSLT 能將 XML 文件轉換成 HTML 網頁。開發 XSLT 是一項與輸出端有高度關聯性的程序，因此不易撰寫。

本論文利用一個稱為 BODE 系統的視覺化工具來製作自動化的 XSLT 產生器。藉由紀錄資料擷取的程序，BODE 系統會學習來源網頁文件與目的 XML 資料文件之間的映對關係，並從這個映對關係計算出 XSLT 程式碼。由於 BODE 系統的資料擷取步驟是以視覺化的方式進行，然後再自動地計算出 XSLT，因此，綜合這兩個步驟就能形成視覺化的 XSLT 產生器。

Automatic Web Page Regeneration Technique Based on BODE Extraction System

Student: Tzu-Chiang Chou

Advisors: Dr. I-Chen Wu
Dr. Lung-Pin Chen

Institute of Computer Science and Information Engineering
National Chiao Tung University

Abstract

XML is a standard way of representing data in a structured format, and being applied for various application domains, such as data exchange, data integration, electronic publishing, and e-Business. XSLT is a language commonly used to transform XML documents, e.g., to transform a XML document to a web page (i.e. HTML file). Since XSLT is tightly related with the representation on the display screen, cooperation between experienced programmer and art designer invokes high cost of developing the XSLT code.

In this thesis, an automatic XSLT generator, based on a visualized web extraction tool, called BODE system, is developed. By extracting the web pages, the BODE system records the mapping from the HTML document of the web page to the XML document which stores the extracted data. From the recorded mapping, the reverse direction transformation, i.e. from the XML document to the HTML document, is derived and outputted as the XSLT code. Since the extraction step is performed by using the visualized tool, and the step of deriving XSLT is automatic, these two steps comprise a visualized XSLT generation system.

誌　　謝

首先我要感謝的是我的指導教授 吳毅成老師，在我碩士的生涯中給了相當多的幫助，並教導我做研究的方法，並適時的給予意見。接著要特別感謝 陳隆彬老師，在我寫這篇論文時，給我相當多的指導和照顧，在多次的討論與研究中，給我寶貴的意見及教導，使得這篇論文能夠順利的完成。

再來要感謝 蘇瑞元學長不斷的給予幫助、輔導，以及他開發的BODE系統，方便資料來源的萃取。還要感謝實驗室BODE組的 秉宏學長、博宇學長、承駿學長、一芳學長、耀興學長、伯鈞同學、益洲同學、文鋒學弟等人平時的照顧，還有我們實驗室碩二的德彥、志祥、仕全這些好伙伴們適時的關心與幫助。

特別感謝我的二姨，當我在台中寫論文時給予非常多的照顧與幫忙，給予我生活上無慮的環境可以專心衝刺論文，還有交大男子籃球隊的大家平時的關心與鼓勵，這些都是我能完成最大的動力。

最後要感謝我的父母、奶奶、弟弟，在我的求學生涯中給予最大的支持與鼓勵，讓我一路走來倍感溫馨。因為有大家的祝福，讓我可以安心的完成學業。謹以此論文，獻給我最親愛的家人們。

目　　錄

表 目 錄

圖 目 錄

第一章 序論

1.1 研究背景

由於科技的日新月異以及日趨成熟的技術，加速了網際網路（Internet）的普及，使得全球資訊網（World Wide Web）已經成為現今人類生活上最受歡迎的傳播媒體之一。隨著網際網路技術的發展，網路上的資訊也以驚人的速度成長。

以 WWW 為例，網頁的資料更新、改版是常發生的事件，因此網站的維護及重整就變成一項相當重要的課題。在網頁上更改資料是一件相當累人的事情，但若是能把資料跟樣式切開，這樣資料維護人員就只要處理資料的更新，而美工人員就單純的改變網頁的樣式，這種方式也成為了現今網站的一種潮流，再搭配著排版的功能就可以要產生出一個網頁。如何保存網頁的樣式跟資料，進而利用這些樣式跟資料來重建出網頁，並讓再生的流程簡單、自動化就是本篇論文的目標。

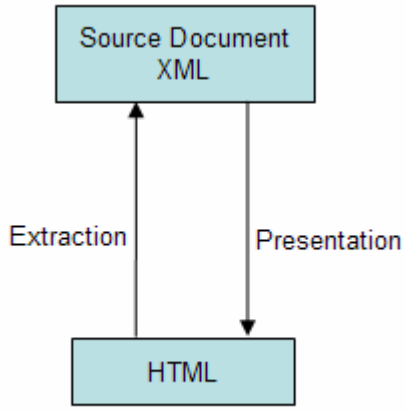


圖 1-1: XML-based 多媒體文件之擷取與呈現

為了整合各個應用領域的多樣化的網路電子文件，在擴展性、網路傳輸、資訊交換、及資料整合管理等各方面都具有優越特性之 XML (eXtensible Markup Language) 乃應運而生。

本論文之主題為研究 XML 文件的“資料擷取”(extraction) 和“呈現”(presentation) 二個功能，並建立自動化的資料擷取與呈現

系統。我們的研究發現，XML 文件的“資料擷取”和“呈現”二個功能在概念上是兩個相反方向的操作。圖 1-1 顯示網頁擷取和網頁呈現之概念圖。

假設有一個 e-learning 系統，擷取多份多媒體文件並整合成 XML-based 教材知識庫，這些教材資訊再轉換成各種形式呈現於學習者眼前，而呈現方式由使用者需求及顯示平台屬性來決定，這些動作基本上可以歸納為以下兩個操作：

- 擷取(Extraction)
  此功能為擷取多個文件之欄位內容，儲存到 XML 文件庫之中。擷取功能對於進階之資訊檢索是非常重要的，大量文件藉由 robot 收集後，以擷取功能達到資料自動化之目的，解決人類不易整理大量資料的問題，例如電子商務網站的資料，便可以利用擷取程式進行分類及檢索等資訊整理的工作。

  在 XML 標準制定初期，擷取與呈現等功能大部份須以人工方式處理，隨者數位內容之各種應用之多元化，目前已有視覺化之擷取工具被開發出來。例如在 BODE 擷取系統中，使用者以滑鼠點選所需欄位，系統會產生 BODE Script（用來紀錄 extraction rule)，之後遇到類似結構之網頁，只要套用 BODE Script 即可自動擷取，新版 BODE 系統更具有隨網頁變化而自動修正擷取程序之功能。

- 呈現(Presentation)
  XSLT 是專為 XML 資料轉換以及呈現所設計之語言，XSLT processor 讀入 XML 文件檔和 XSLT 檔，就可以產生目的文件，例如，在 e-learning 教材撥放系統中，XML 格式之教材內容透過 XSLT 轉換機制轉成 HTML，搭配 Flash、JavaScript 等多媒體網頁技術，學生可以透過瀏覽器觀看互動式的教學內容。

1.2 研究動機與目標

開發 XSLT 是一項與輸出端有高度關聯性的程序，若沒有視覺化介面的輔助，撰寫 XSLT 是一項耗時的工作，例如若沒有教材編輯器供使用者製作畫面樣式，程式設計師很難直接撰寫 XSLT 來輸出想要之樣式。

擷取程序已有一些視覺化工具可以達成，例如 BODE 系統(Browser-Oriented Data Extraction System)[6][10]，它結合 GUI (Graphic User Interface) 的圖形介面應用程式，以 BODE 查詢語言為基礎，使用者透過圖形介面之操作，系統自動建構出相對應的 BODE 查詢語法架構，利用所產生的 BODE Script，使用者能自動化的萃取出網站上的資訊，並將資料存成 XML 格式，做更進一步的資訊處理與整理。

樣式就是網頁的呈現方式，大部分的 HTML 標籤都是用來做樣式的呈現，搭配 XSLT 我們可以將 XML 資料轉化成我們想要的 HTML 網頁樣式。但 XSLT 不易撰寫，且較少有 GUI 介面的工具來開發 XSLT。

本論文將研發自動化的 XSLT 產生工具，來達到自動化網站再生的目地。本論文的主要概念是以 BODE 系統為工具，先進行資料擷取，藉由紀錄資料擷取的程序，該工具會學習原始網頁檔案與目的資料檔案之間的映對關係，進而計算出 XSLT。

雙向 XML 轉換元件具有以下優點

- 基於現有擷取工具來發展呈現工具，縮短開發時程。
- 以輸出樣式之畫面作為 XSLT 產生之依據，而非由 XSLT 程式產生輸出樣式，符合人類習性。
- 擷取與呈現程序對照進行，增加正確性。

1.3 論文架構

在第二章我們介紹 XPath、BODE，以及 XSLT 等相關背景；第三章則介紹主要演算法；第四章將著重在系統之開發與實作細節；第五章是整篇論文之總結，以及未來之擴充性探討。

第二章 背景

在 2.1 節會介紹 XML 與 XPath(**XML Path Language**)，而 2.2 節是說明 BODE(Browser-Oriented Data Extraction System)系統的功能及其特性， 2.3 節則介紹 XSLT(eXtensible StyleSheet Language Transformation)轉換語言。

2.1 XML 與 XPath

在 2.1.1 節會從 HTML 介紹到 XML 的起源與特性，而 2.1.2 節是說明 XPath 的功能及語法。

2.1.1 HTML 與 XML 介紹

回顧全球資訊網所採用的 HTML (HyperText Markup Language)網頁標準格式。HTML 基本上是一個以 SGML(Standard Generalized Markup Language)所定義出來的描述語言(Markup Language)。 然而，HTML 卻有下列不足的地方：

- 擴充性 (Extensibility) ：HTML 並無法讓使用者可以自行定義標籤(tag)、屬性(attribute)。
- 結構化 (Structure) ：HTML 的標記只是註記，並不支援正規結構的定義，以表示資料庫(database)中的 schema 或者是物件導向的 hierarchy。
- 驗證 (Validation) ：HTML 也無法透過 HTML 語言的定義，來為某個 HTML 文件進行查核驗證(Validation)的工作。

除了彌補 HTML 的不足，為了滿足各式各樣的應用需求並且將 Web 的發展推向一個分散式資料處理的新領域，W3C(Word Wide Web Consortium)以及其參與成員於是制訂了 XML(eXtensible Markup Language)。

相對於 HTML 網頁格式， XML 的主要優點是使用者可以自行定義所需要註記之語言。這讓使用者得以保存所註記資料的語意及與將其資料結構化。舉例來說，一家商店可能保留其顧客的基本資料。假設該商店的保留的某顧客資料如下：

周大強
電話：(03)5712121
地址：新竹市大學路 1001 號

若是使用 HTML 來儲存，一個可能的寫法如下：

```
<H1>周大強</H1>
<TABLE BORDER=”1”>
<TR><TD>電話：</TD>
<TD><B>(03)5712121</B></TD>
</TR>
<TR><TD>地址：</TD>
<TD><B>新竹市大學路 1001 號</B></TD>
</TR>
</TABLE>
```

然而，對這份 HTML 資料文件而言，它要對 Web 所顯示的真正資訊—顧客姓名、地址與電話—已經消失；相對地，它只顯示了在文字上，哪些要變成標題，哪些要製成一個 table，哪些要成為粗體字等等。這文件中的名字、電話、地址等語意已經不見了。若改用 XML 格式，則可將這些語意存於標記（tag）中。例如：

```
<PERSON>
<NAME>周大強</NAME>
<PHONE>(03)5712121</PHONE>
<ADDRESS>新竹市大學路 1001 號</ADDRESS>
</PERSON>
```

上述的例子,顯示 XML 格式的資料仍可以保留 NAME、PHONE、ADDRESS 等訊息，這可用於與其他資料模式的系統（如資料庫）做資料交換。這是 XML 資料模式重要的地方。至於呈現的部份可用 XSLT (eXtensible Stylesheet Language Transformation)來轉化成網頁。

簡言之，XML 不但是用來讓機器之間交換資料的橋樑，還可以是人與人之間的溝通媒介；XML 也不僅僅與 Web 有關，更可以和所有一般的事務相關。XML 已經是當今電子商務資訊交換(Information Exchange)的標準。由於資料使用的重複率高，將資料改用 XML 儲存，對於資料的保存及再利用都有很好的效果，因此 XML 應該會在極短的未來廣泛的被採用，漸漸的取代 HTML 的地位。

2.1.2 XPath (XML Path Language):

XPath 是 W3C 的一種標準[1][20][21]，他所定義的語法和語意可供 XSLT 使用，主要目的是用來定出 XML 文件中各個部分的位置。XPath 看到的是 XML 文件的邏輯結構，而非實體結構。XPath 借用 URL 的路徑觀念，從而得以在 XML 文件的階層結構中來回遊走。

XPath 主要是由一個取向軸(Axis)一個節點測試(Node-test)及零至多個謂語(Predicates)所構成：

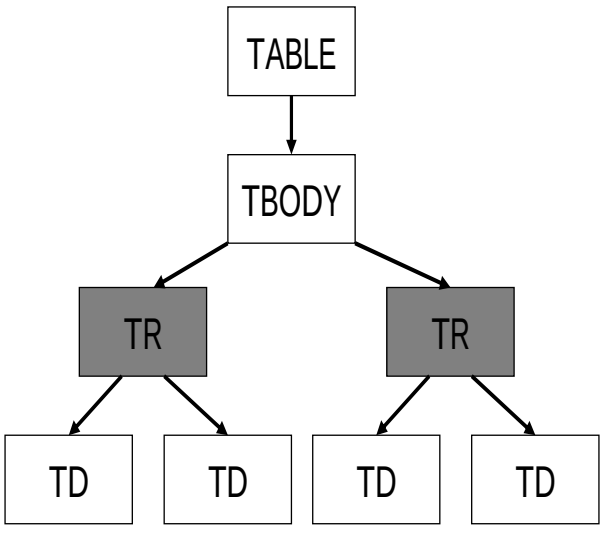
- 取向軸(Axis)：指定要選取的節點和目前節點在結構樹上的關係。
  - ancestor, ancestor-or-self
  - Parent, self, child
  - descendant, descendant-or-self
  - following, following-sibling
  - preceding, preceding-sibling
  - attribute
  - namespace

- 節點測試(Node-test)：判定節點是否符合條件。
  - Nodename
  - text()
  - comment()

- processing-instruction()
- node()

◆ 謂語(Predicates)：謂語可有可無，也可以有好幾則。我們可以在謂語中使用運算式來篩選所選取的節點。
  - ■ 謂語運算式：
    - Boolean: or , and , = , != , <= , < , >= , >
    - Numeral: + , - , * , div , mod
    - Functional: position( )
  - ■ 範例:
    - child::tr[position( ) > =0]
      → 取 child 位置大於 0 的
    - child::table[attribute::width = “150”]
      → 取 child 名稱是 table，且有個屬性 width 的值是 150。
    - tr[position()=0 AND child::td]
      → 取第 0 個 tr，且有 child 為 td。

圖 2-1 是 XPath 的一個範例，目前節點是/TABLE/TBODY，他的取向軸是"child"，節點測試是"TR"，謂語是"position()>=0"，意思是只要是 TBODY 的 child，且位置大於等於 0 就會被選取。因此圖中的那兩個 TR 就都被選出來了。



.../TABLE/TBODY/child::TR[position( ) >= 0]

圖 2-1: XPath 的範例

2.2 BODE 系統(Browser-Oriented Data Extraction System)

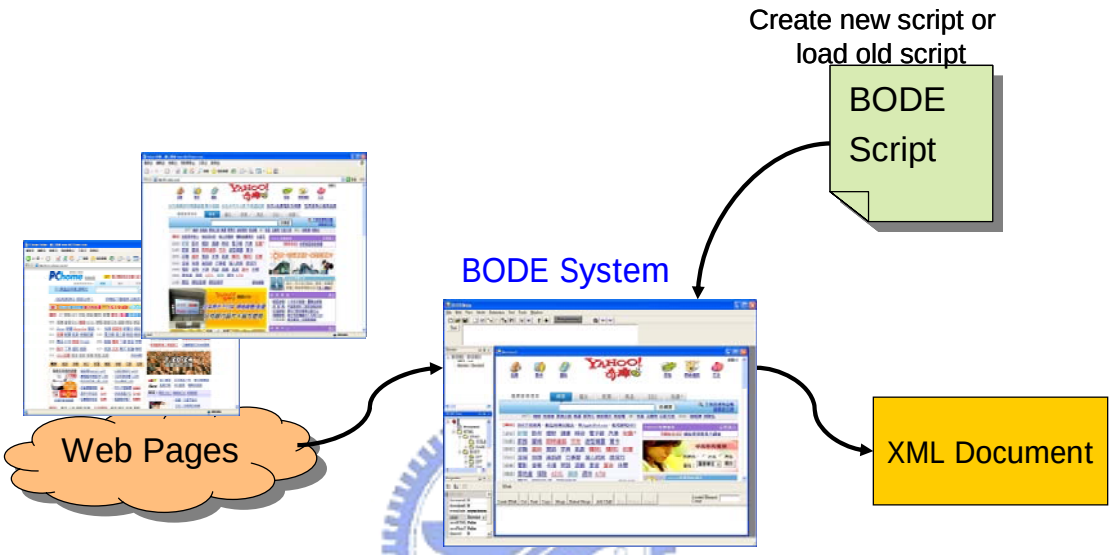


圖 2-2: BODE System 萃取網頁示意圖

BODE[6][10]是一套具有視覺化介面的資料萃取系統，採用 W3C 制定的 XPath 來定位所要萃取的資料在文件中的位置，並實作 BODE 之 Web 文件萃取引擎，及視覺化的操作介面。

BODE 語言與 WIDL[24]語言較為類似，是屬於服務（service）或程序(procedure)為基礎的萃取模式。WIDL 將原本用於物件導向的 IDL 介面定義技術導入 WWW 文件萃取中，其定義的 Services，相當於程式語言中的程序。將網頁的內容對映到程式的變數中，以作為處理的對象。

BODE 的語法是一套以 XML 為基礎的語法，因此整個 BODE script 的格式為 XML。以下是一個簡化的 BODE 的 script，這個例子將說明 BODE 如何的萃取一個網站的資料。

```
<BODE>
    <INIT URL= "http://www.yahoo.com.tw"
        SERVICE="HomePage"... />
    <SERVICE NAME="HomePage">
        <VAR NAME="CatURLs" XPATH="..." .../>
        <FOREACH NAME="CatURL" FROM="CatURLs" ...>
            <INVOKE SERVICE="Category" URL="$CatURL"/>
        </FOREACH>
    </SERVICE>
    <SERVICE NAME="Category">
      ...
    </SERVICE>
</BODE>
```

- BODE script 最外層是<BODE>標籤。<BODE>標籤中，有一個<INIT>標籤以及一或多個<SERVICE>標籤。

- 每一個<SERVICE>標籤中的內容是用來描述如何萃取某一類的網頁。<SERVICE>標籤可以包含<VAR>、<FOREACH>以及<INVOKE>標籤。

- <INIT>標籤則描述這個 BODE script 是由哪個網址開始載入網頁。並且指出這個初始的網頁是由哪一個 SERVICE 標籤中的敘述來處理。

- <VAR>標籤是指一個變數。變數的名稱紀錄在 Name 的屬性中。而要萃取的資料所在位置的結構則以 XPATH 表示。在名為 HomePage 的 Service 標籤中，要萃取一項資料並放在變數 CatURLs 中。

- <INVOKE>標籤是執行超鏈結或是模擬滑鼠的點選動作,可以用來連接到下一個網頁，或是點選畫面上的按鈕以執行網頁中的 script 程式。若是要連接到下一頁，則 SERVICE 屬性中要指定由哪一個<SERVICE>標籤中的內容來處理<INVOKE>連接到的網頁。

<FOREACH>標籤是指一個迴圈，是依據 FROM 屬性中指定的 CatURLs 變數中取出一個值成為 CatURL，並執行<FOREACH>標籤中所指定的動作。

在這個 FOREACH 中的<INVOKE>標籤的 XPATH 屬性內容為 CatURL 變數所選取到的物件，因此<FOREACH>會針對 CatURL 變數中的每一個超鏈結連接到下一頁。

根據 BODE Script 裡 to 的屬性，我們可以把萃取到的資料存成結構化的 XML 文件，如圖 2-3 所示，網頁上每個 row 都是一本書，因此我們 Foreach 每個 position>0 的 row，給個 to=”Book”，裡面用四個 Var 來取 TD[0]~TD[3]，分別 to 到 Title、Author、Price、Year，如此便可將資料以結構化的方式儲存在 XML 文件中。

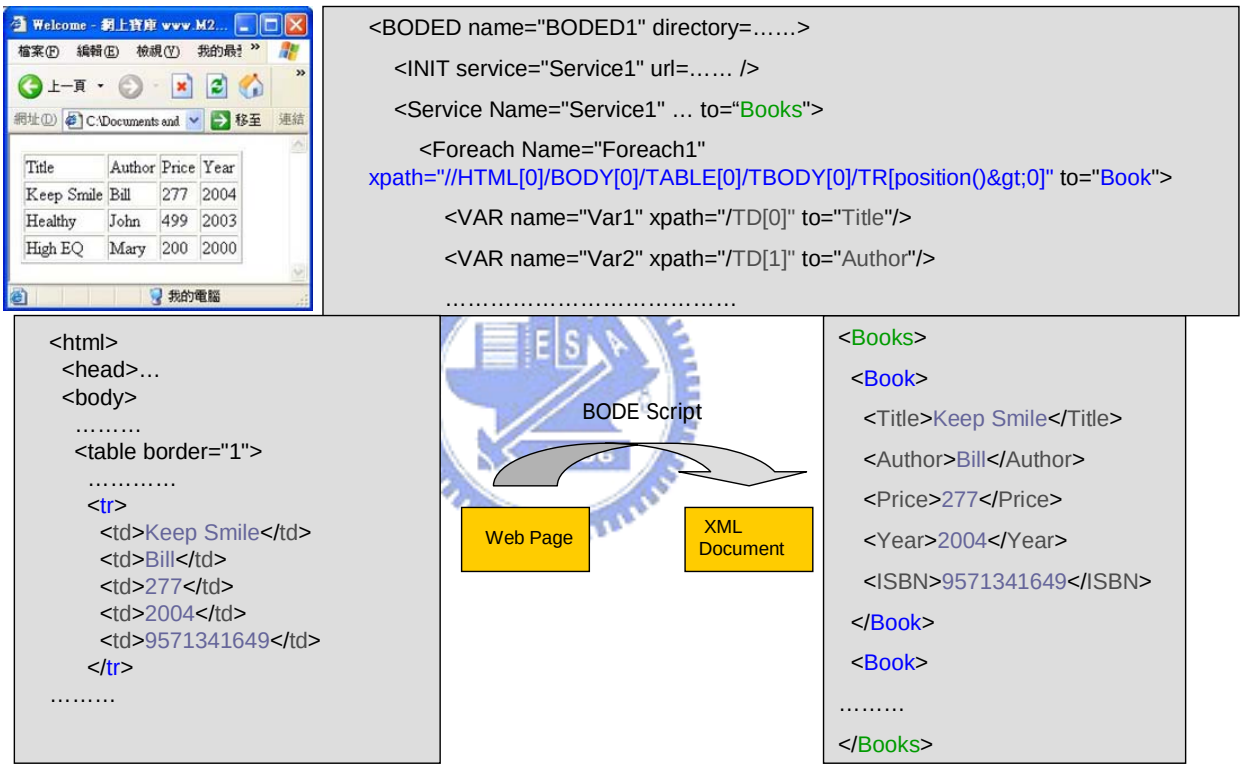


圖 2-3: BODE 萃取網頁資料到 XML 文件的範例

BODE 系統的特點有:

- Navigate with GUI
- Conform to browsers
- Locate elements with XPath
- Fill Form
- Simulate clicking
- Plug-in Programs

在這邊我們將 BODE 與其他的萃取系統如 XML-QL、XQuery、W3QL、WebOQL、WIDL、Lixto，做個摘要式的分析比較。

	XML-QL	XQuery	W3QL	WebOQL	WIDL	Lixto	BODE
可處理的文件格式	XML	XML	HTML + XML	HTML + XML	HTML + XML	HTML + XML	HTML + XML
變數萃取的格式	DOM-like	XPath	PERL code	DOM-like	DOM-like	Elog (Prolog-like)	XPath
可適用於須 Cookie 等之網頁	No	No	No	No	Yes	Yes	Yes
填表能力	No	No	Yes	No	Yes	Yes	Yes
整合外掛程式的能力	None	None	PERL code	None	IDL for other programs	No	Plug-ins
模擬按鍵能力	No	No	No	No	No	No	Yes
與 Javascript 結合的能力	No	No	No	No	No	No	Yes
提供視覺化工具	No	No	Only for FORM	No	No	Only for Extract Condition	Yes

表格 2-1: BODE 與其他資料萃取系統的比較

2.3 XSLT 轉換語言

在網頁如此普及的現代，如何將比較嚴謹的 XML 語法改用 HTML 做呈現變成一個很重要的課題，而 XSLT (eXtensible StyleSheet Language Transformation)的推出可以滿足我們的需求。XSLT 是 W3C 的一種標準[1][22][23]，最主要的目的就是用來轉換 XML 資料，我們可以將 XML 資料透過 XSLT 轉換文件結構，輸出成另外一個 XML 資料，或是改用網頁形式輸出，現在的瀏覽器大多都提供 XSLT 的支援。傳統的資料庫有標準 SQL 語言可供操控設計好的資料庫；而如今的 XML 文件則有標準 XSLT 轉換語言可與之匹敵。和關連式資料庫(relational database)相比擬，XSLT 的確可以稱做是 XML 的 SQL 語言。圖 2-4 是 XSLT 處理器的處理流程，XSLT 先處理 XML 及 XSLT 的樹狀結構，經過結構的轉換產生 Result Tree，最後再輸出結果。

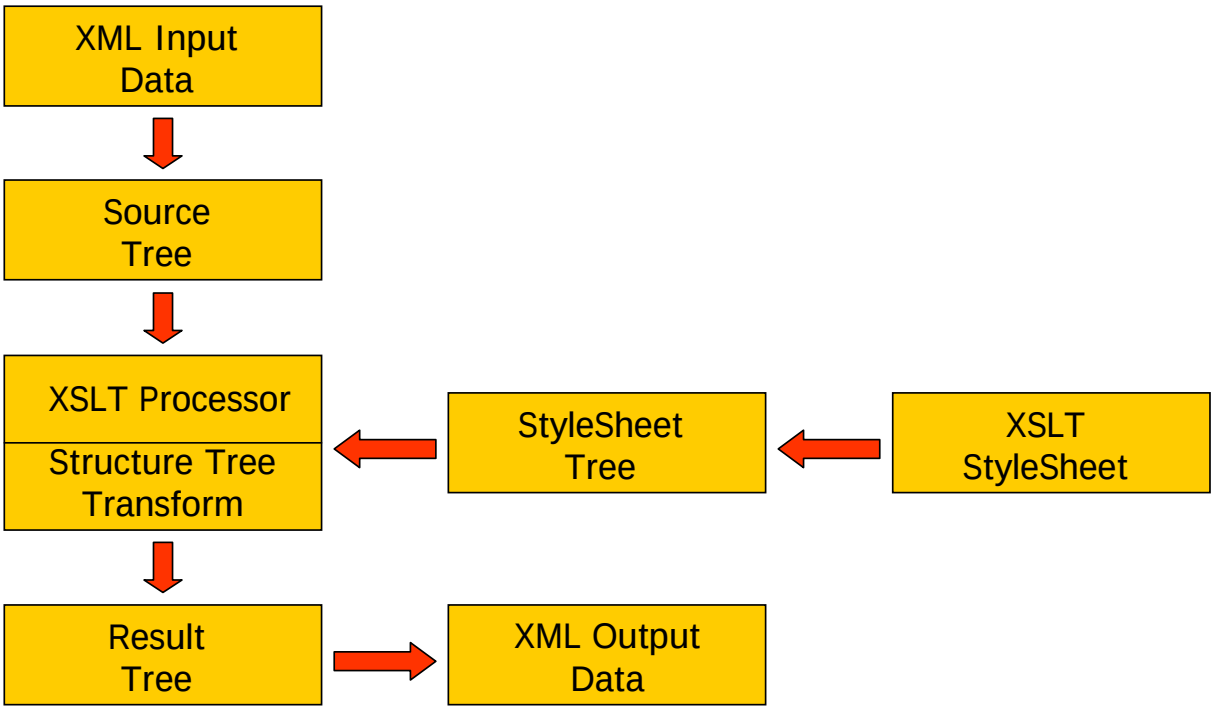


圖 2-4: XSLT 處理器的處理流程

除了 XSLT 之外，CSS 也可以把 XML Document 轉成 HTML 方式來做呈現。

- CSS(Cascading Style Sheets) 使用簡單的比對，然後直接將設計者所設定的顯示屬性值使用在文字區塊上，因此完全沒有改變原始文件的結構及內容順序。
- XSLT (Extensible Stylesheet Language Transformation) 提供了轉換的機制，可以隨意的比對原始文件的結構及選擇其內容，以建構出全新的文件。表格 2-2 是 XSLT 和 CSS 的一些比較：

	CSS	XSL
可以使用在 HTML 檔中嗎?	可以	不可
可以使用在 XML 檔中嗎?	可以	可以
是個轉換語言嗎?	不是	是

表格 2-2: XSLT 和 CSS 的比較

對 XSLT 來說，文件是節點構成的樹狀結構，他也認得多種節點。
表格 2-3 是 XSLT 能辨識的 XML 節點類型和 XSLT 處理器對他們的處理方式：

節點	說明
文件根節點(Document root)	文件的起點
屬性(Attribute)	存放屬性值，周圍的空白會先被去掉
註解(Comment)	存放註解的內容，不包含「<!--」及「-->」
元素(Element)	內含元素裡的所有字元資料，包含所有子代元素的字元資料
命名空間(Namespace)	存放命名空間 URI
處理指令(Processing instruction)	存放處理指令的文字，不包括「<?」和「?>」
文字(Text)	存放節點裡的文字

表格 2-3: XSLT 能辨識的 XML 節點類型及說明

第三章 再生樹與自動化 XML 文件轉換

3.1 資料擷取和資料呈現

XML 文件的“資料擷取”(extraction) 和“呈現”(presentation) 二個功能在概念上是兩個相反方向的操作。明確地說，資料擷取是以 BODE Script 來記錄擷取的程序，而呈現則是以 XSLT 來指定 XML 資料如何轉換成 HTML 網頁。圖三顯示 HTML 網頁 H 和 XML 文件 X 的轉換關係：H 經過 BODE-Script BS 轉換成 X，而 X 藉由 XSLT XS 轉成 H，此關係可以下列邏輯式子來表示：

$$H \otimes BS = X, \text{ and} \qquad (1)$$

$$X \oplus XS = H$$

由公式(1) 可以聯想到，若是 H、BS、和 X 均為已知，那麼 XS 就可以被推導出來。

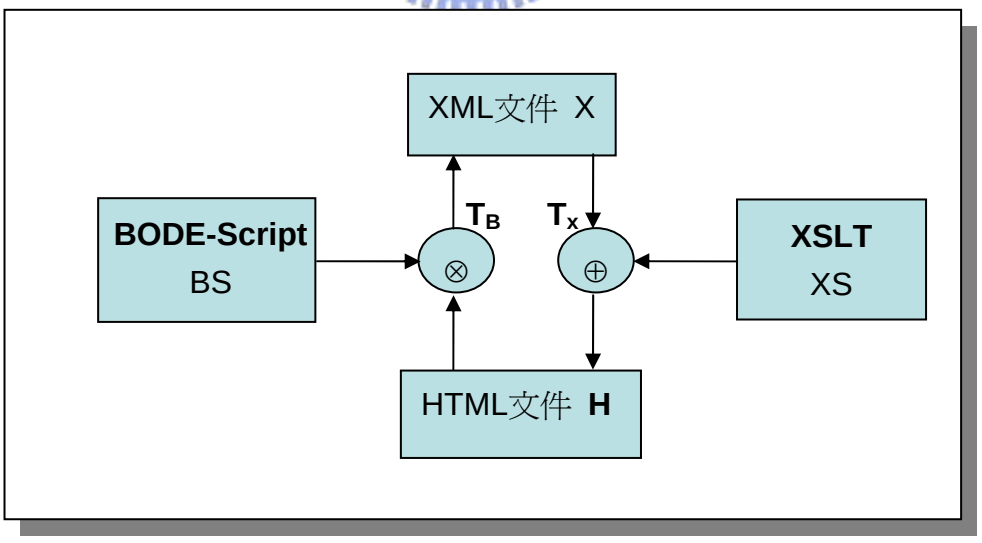


圖 3-1: HTML 文件與 XML 文件的結構的轉換與計算

3.2 再生樹 R-Tree

在公式(1) 與 圖 3-1 之中，在資料擷取時，必須先給定一個樣本網頁 *h*，作為第一次資料擷取之用。樣本網頁 *h* 將主導 XSLT 文件 *XS* 之輸出樣式，也就是說，XSLT 所轉換出來的文件 H 與 sample 文件 h 類似。例如，在 sample 文件 S 中，商品資料只有一筆，而在結果文件 H 中，商品資料會有十筆，因為資料庫中有十筆紀錄。為了表示 sample 文件 S 如何擴展成結果文件 H，我們的做法是將 sample 文件 S 擴展成一個再生樹 (Regeneration tree, R-Tree)。R-Tree 的節點中加上了一些屬性來描述 S 如何擴展成結果網頁。

R-tree 是一棵樹，其詳細定義如下：

定義 Regeneration Tree (R-Tree)

- **R-Tree** (Regeneration Tree) 之結構與 sample HTML document 相同。R-Tree 中的節點都稱為 R-Node。

- **R-Node** 包含以下三個屬性：
  - **R-Tag** (Regeneration Tag)
    這個節點的 HTML Tag。

  - **R-DataSource** (Regeneration Data Source)
    R-DataSource 是一個 XPath，指向 XML 文件中的節點，也就是這個節點的資料來源。

  - **R-Pattern** (Regeneration Pattern)
    - 以類似 DTD 的語法定義出子節點的結構。
      (1) Operands: '0', '1', '2', … 分別代表 child 0, child 1, child 2, …。

      (2) Operators 定義如下:
        - a,b => node *a* 串接 node *b*。
        - a^* => node *a* 重複直到來源資料用完。
        - a^n => node *a* 重複 n 次或是直到來源資料用完。

- a | b => child *a* 或是 child *b*，依照資料順序決定。

以下是 R-tree 的一些例子。

範例一：

範例一是雅虎奇摩[5]首頁上面的天氣預報。網站每天要維護的資料只有溫度的部份，其他樣式的地方都不會改變，因此我們可以利用 BODE 萃取系統將氣溫的地方萃取下來，之後就只要在 XML 資料中改變溫度的值，則 HTML 中對應的溫度資料就跟著改變，圖 3-2 顯示 HTML 與 XML 的對應關係。

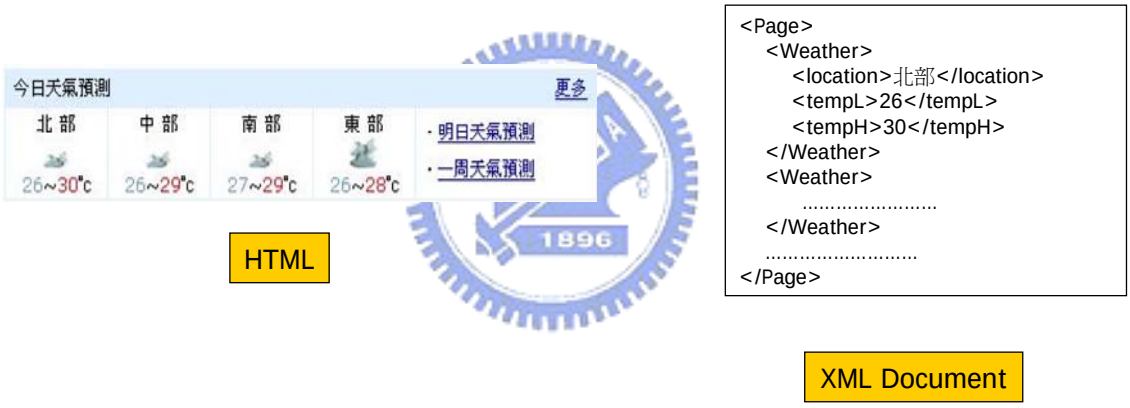


圖 3-2: Yahoo 天氣 sample 與 XML 對照圖

圖 3-3 是 BODE Script 萃取的結果，裡面記載了 From 跟 To 的屬性對應關係，From 就是網頁的 XPath ，To 的值是存到 XML 中的 XPath ，利用這個關係，我們可以在 Regeneration Tree 中給定 R-DataSource，也就是 To 的值。

因為這個範例是一對一的對應關係，所以 R-Pattern 就是 child 各出現一次，而 R-Tag 就是原本網頁相對應節點的 tag。圖 3-4 就是這個範例最後建出來的 Regeneration Tree。

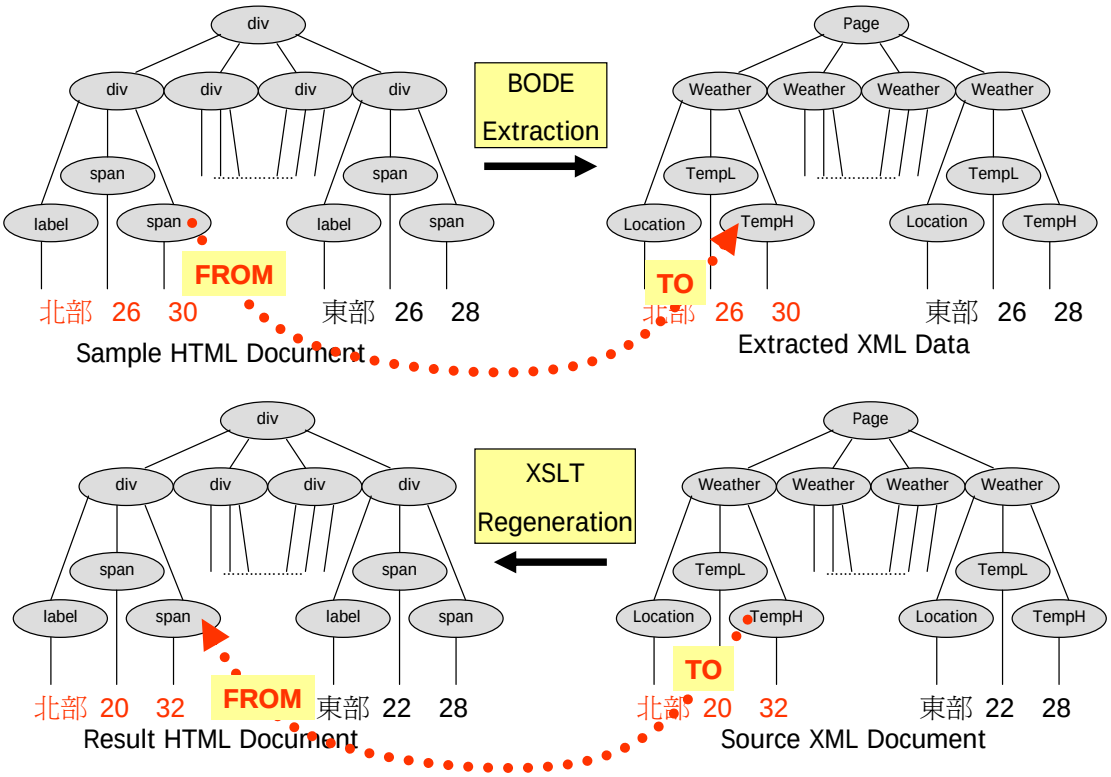


圖 3-3: 範例一中的 FROM-TO 對應圖

R-Node

- R-Pattern = (child 0) (child 1)...
- R-DataSource = to-string
- R-Tag = HTML Tag

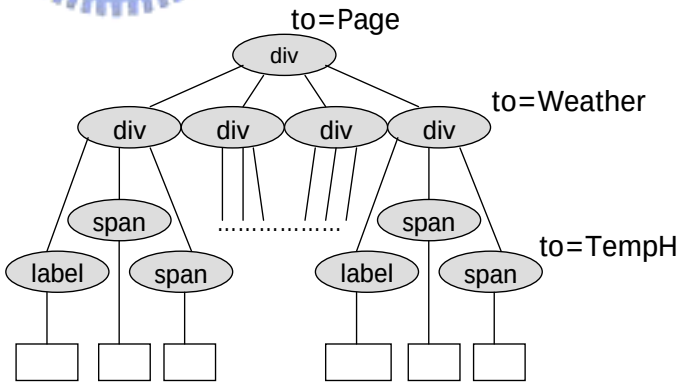


圖 3-4: 範例一的 Regeneration Tree

範例二:

範例二是一份書籍的例子，這範例是要顯示一列 Title 列後，後面都照紅色、綠色、紅色、綠色、…的方式去重複樣式，直到資料結束，圖 3-5 顯示這範例 HTML 與 XML 的對應關係。

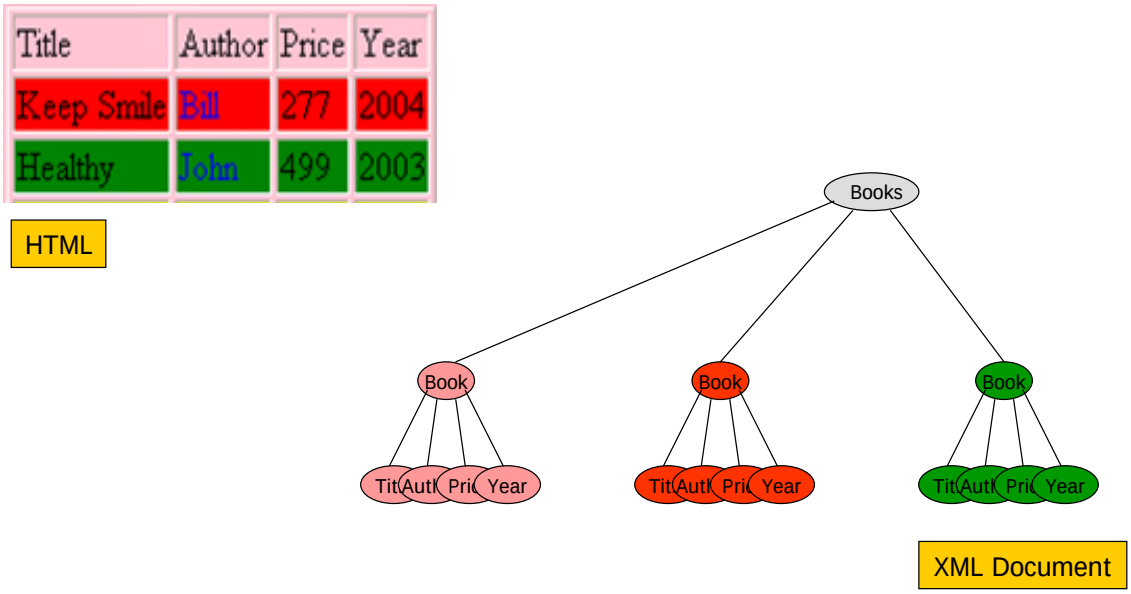


圖 3-5: 書籍 sample 與 XML 對照圖

圖 3-6 顯示這範例 BODE Script 裡 From-To 的對應圖，每個 tr 都萃取到一本 Book 中，當資料庫的 Book 資料新增後，也可以套用產生出來的 XSLT 把資料都呈現出來。

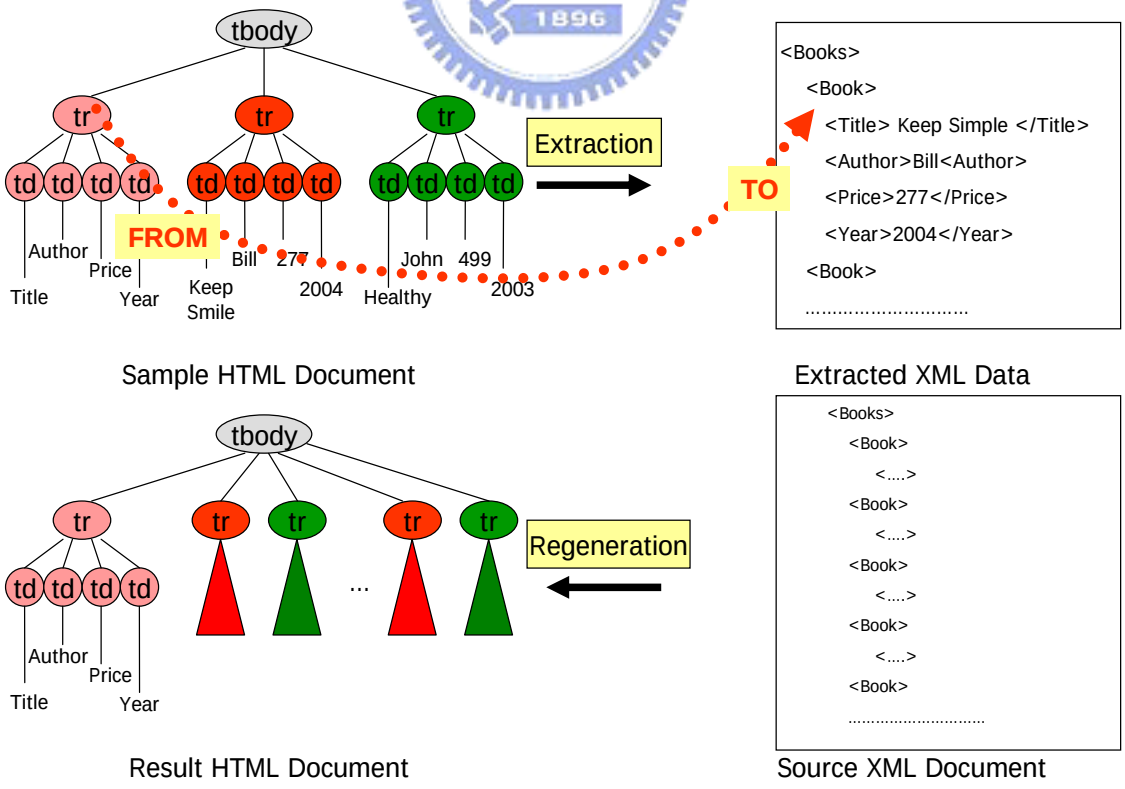


圖 3-6: 範例二中的 From-To 對應圖

圖 3-7 顯示這範例的 Regeneration Tree，我們可以看到 tbody 的地方有個 P=(0,(1,2)^*)，他就表示要 tbody 下的第一個 child 出現一次(Title 列)，而之後是第二跟第三個 child 一直重複交替出現，也就是資料依序用紅色、綠色的樣式來輪流呈現，直到資料結束。

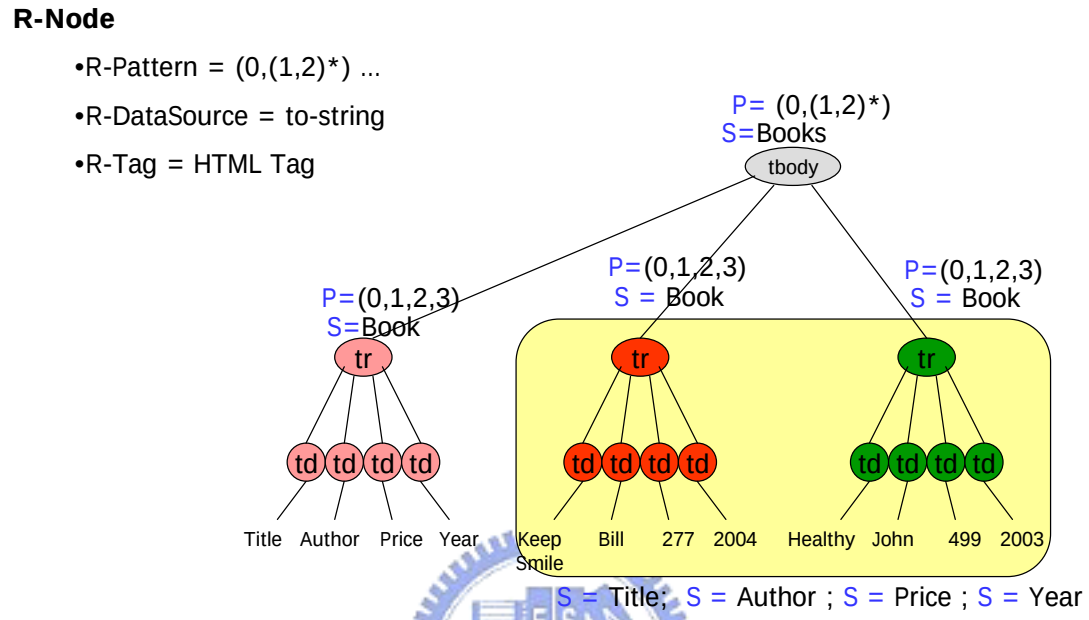


圖 3-7: 範例二的 Regeneration Tree

範例三：

這是一份電子技術手冊(ETM – Electronic Technical Manual)的範例。在 XML 中，一個 Chapter 下有很多個 Section，而每個 Section 底下可能有 Title、Para 及、Caution 這三種 child，個數及出現的順序都不固定，但呈現時希望照這三種資料出現的順序來做呈現。

圖 3-8 是 ETM 網頁資料與 XML 的對應關係。

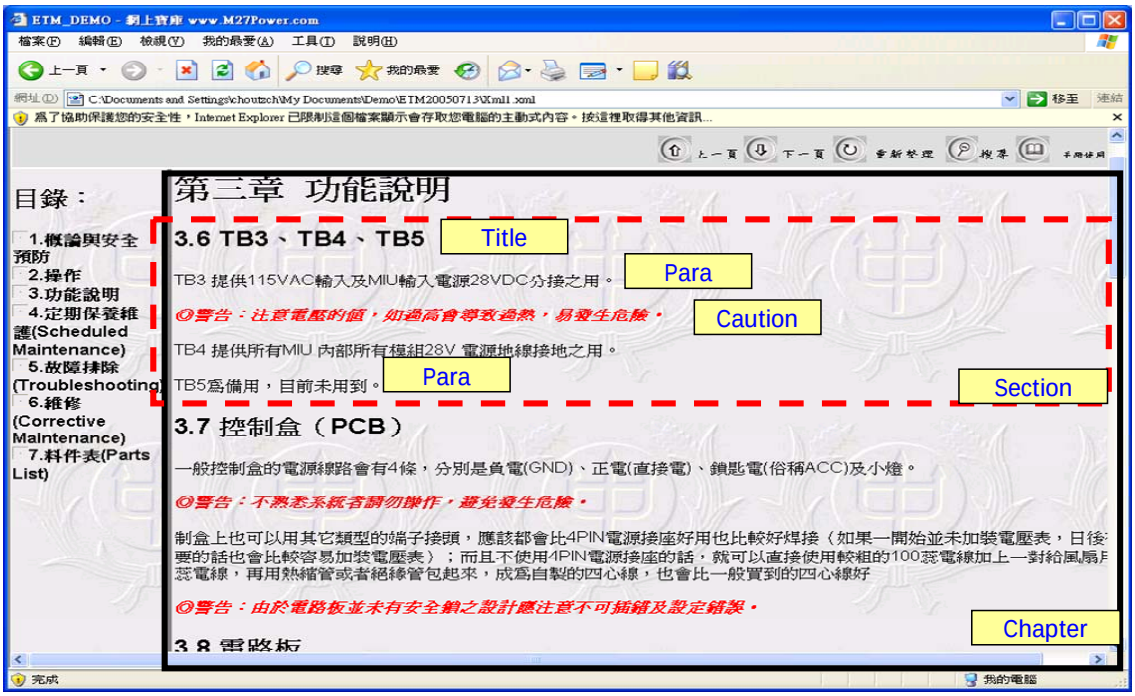


圖 3-8: ETM 的 sample 與 XML 對照圖

圖 3-9 是 BODE 系統萃取及 XSLT 再生的示意圖，<H3>對應到 Title，<p>對應到 Para，而<H4>對應到 Caution。當資料庫的資料為 Title、Para、Caution、Caution、Para 時，用 XSLT 還原回來也會依<H3>、<p>、<H4>、<H4>、<p>的順序來做呈現。

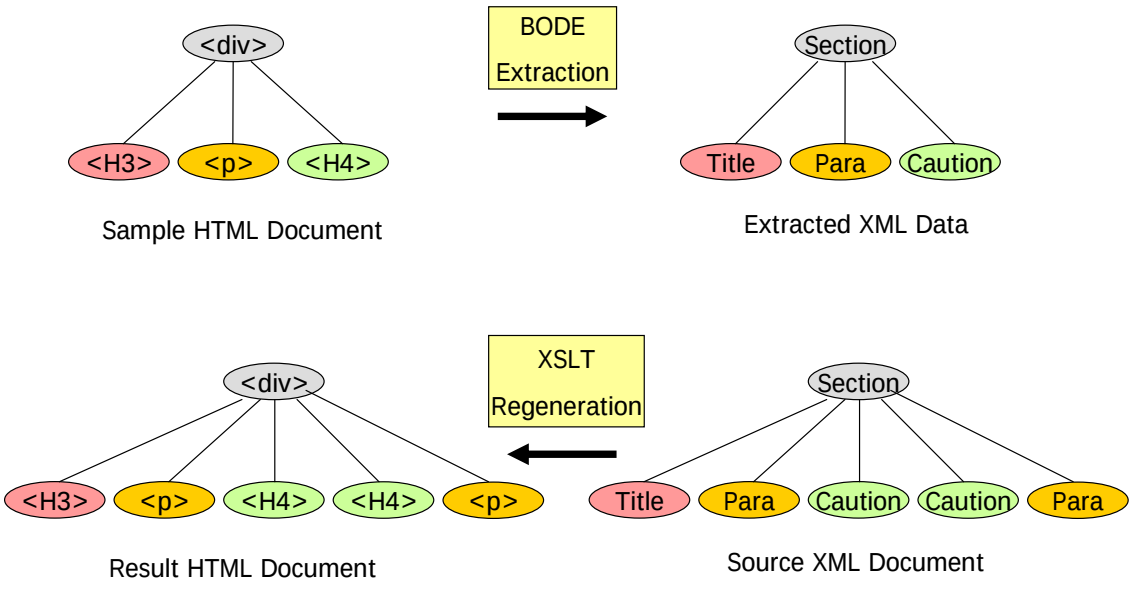


圖 3-9: 範例三中的 From-To 對應圖

圖 3-10 是這個範例的 Regeneration Tree，在 div 這個 R-Node 中的 R-Pattern 是((0|1|2)^*)，表示依照資料來源的型態來選擇對應的 child 0，child 1，或是 child 2，並且重複直到資料來源用完為止。

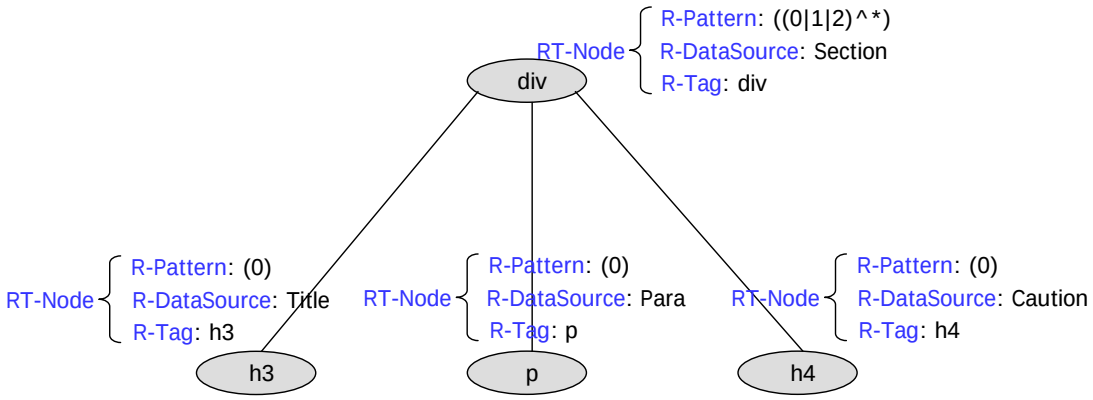


圖 3-10: 範例三的 Regeneration Tree

3.3 使用 R-Tree 產生 XSLT

自動化產生 XSLT 之關鍵是 R-Tree。R-Tree 基本上是 sample HTML document S 再加上一些屬性來描述 S 如何擴展成結果網頁。使用 R-Tree 產生 XSLT 的步驟如下：

1、將 R-Tree R 初始化成與 sample HTML document S 相同。注意此時 R 中每個節點 r 的 r.RTag 是由 sample S 中複製而來，r.RPattern 則由使用者輸入。

2、呼叫 MarkRTree()函數來設定 R 中每個節點 r 的 r.DataSource 屬性。r.DataSource 是一個 XPath，指向 XML document 之節點，表示 R 中的節點 r 的資料來源。MarkRTree()函數之虛擬碼請見表格 3-1。

3、將 R-Tree R 進行遞迴式的展開，產生 XSLT 檔案。R 中每個節點 r 的 r.RPattern 記錄著 r 的子節點應該如何展開，展開的規則請見 R-Tree 之定義。節點 r 的 r.DataSource 記錄著 r 的子節點的資料來源為何。詳細流程請見表格 3-1 之 GenerateXSLT()函數之虛擬碼。

```
Algorithm AutoGenerateXSLT(MSHTML S, XMLDocument BS)  {
  MSHTML S = sample HTML document;
  XMLDocument BS = BODE Script to extract S into XML data;

  //Step 1:  Init R-Tree as the sample HTML document
  XMLDocument R = S;
  //Step 2:  Mark R-Tree
  R = MarkTree( R, BS.root() );
  //Step 3:  GenerateXSLT
  XMLDocument XS = GenerateXSLT(R.root());
  //Algorithm finished!
  return the result XSLT document XS;
}

// Mark R-Tree
XMLDocument MarkRTree(XMLDocument R, XMLNode r) {
  X = EvaluateXPath(r.from);
  foreach p in X do
    let p.RDataSource = r.to

  foreach child node i of r do
    MarkRTree(R, r.child[i]);
}

// Generate XSLT
XMLDocument GenerateXSLT(XMLNode r) {
  N = the number of children of r;
  TemplateNode tn[N];
  foreach child node i of r do
    tn[i] = GenerateXSLT(r.child[i]);

  Create TemplateNode tn_of_r for node r;
  tn_of_r consists of child templates tn[] and controls the repeat pattern of
    these templates according to r.RPattern;
}
```

表格 3-1: 使用 R-Tree 產生 XSLT 的演算法

4. 實作結果

在 4.1 節中介紹整個系統的開發環境與流程，而 4.2 節是以實例來說明此系統如何應用。

4.1 開發環境與系統架構

在 4.1.1 節中是說明整個系統的開發環境，而 4.1.2 節是介紹此網頁再生系統的流程與架構。

4.1.1 開發環境

這套系統是以 Microsoft Visual Studio C#.Net 2003 來開發的，其中主要用到的函式庫如下：

- Microsoft.mshtml
  - HTMLDocumentClass
  - IHTMLElement
- System.Xml
  - XMLNode
  - XMLDocument
  - XMLElement

此外還用 HTML-Kit 這套工具來做驗證 XML 格式正確性的工作。

4.1.2 流程及系統架構

整個網頁再生的流程如下:

Step1: 先將 sample page 透過 BODE 系統萃取，產生出 BODE Script 及 XML 資料。

Step2: 使用者在 sample page 中輸入 R-Pattern，再將 sample page 和 Step1 產生的 BODE Script 傳給這套 XSLT Generator 系統，產生出可再生網頁的 XSLT。

Step3: 將 XML 資料及剛剛產生的 XSLT 透過 XSLT 處理器，即可再生出跟原 sample 網頁樣式相同的網頁。

圖 4-1 是整個系統的流程圖。

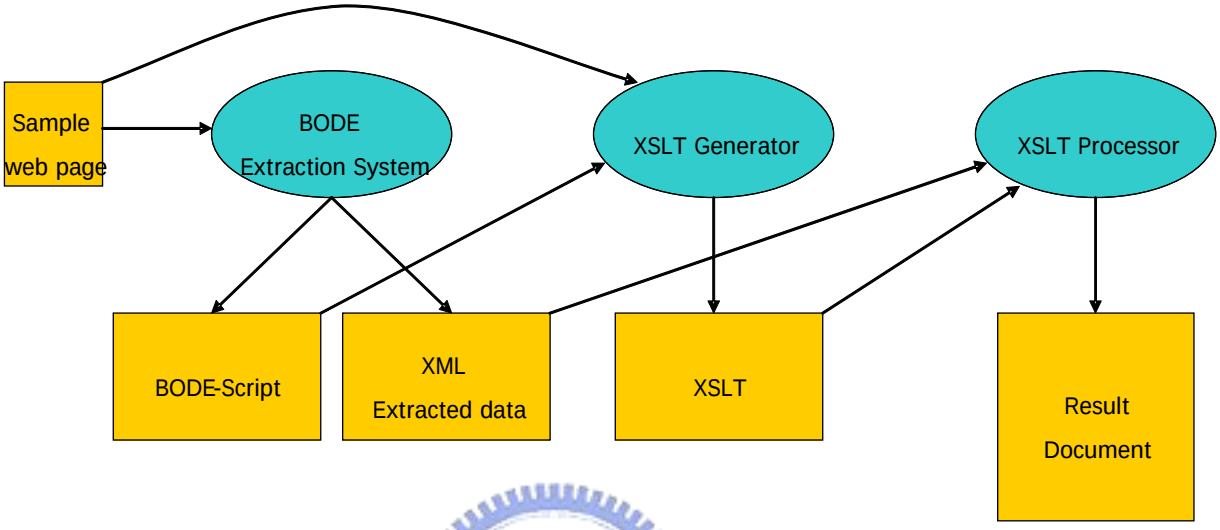


圖 4-1: 網頁再生流程圖

4.2 實作成果

4.2.1 節及 4.2.2 節是以兩個不同實例來說明此系統如何應用。

4.2.1 應用實例一

我們第一個 Demo 的例子是將此網頁再生系統套用在電子技術手冊(ETM－Electronic Technical Manual)上。圖 4-2 是一個 ETM 的 sample 網頁。在 XML 資料中 Chapter 下有 Section，Section 下有 Title、Para、Caution，且希望照資料中 Title、Para、Caution 出現的順序來做呈現。

XML 資料與 HTML 網頁的關係是:Section 對應到<DIV>，Title 對應到<h3>，Para 對應到<p>，Caution 對應到<h4>，因此 DIV 我們要給定 R-Pattern 為((0|1|2)^*)，這樣才能照資料中 Title、Para、Caution 出現的順序來做呈現。此 sample 網頁部分 HTML 如表格 4-1:

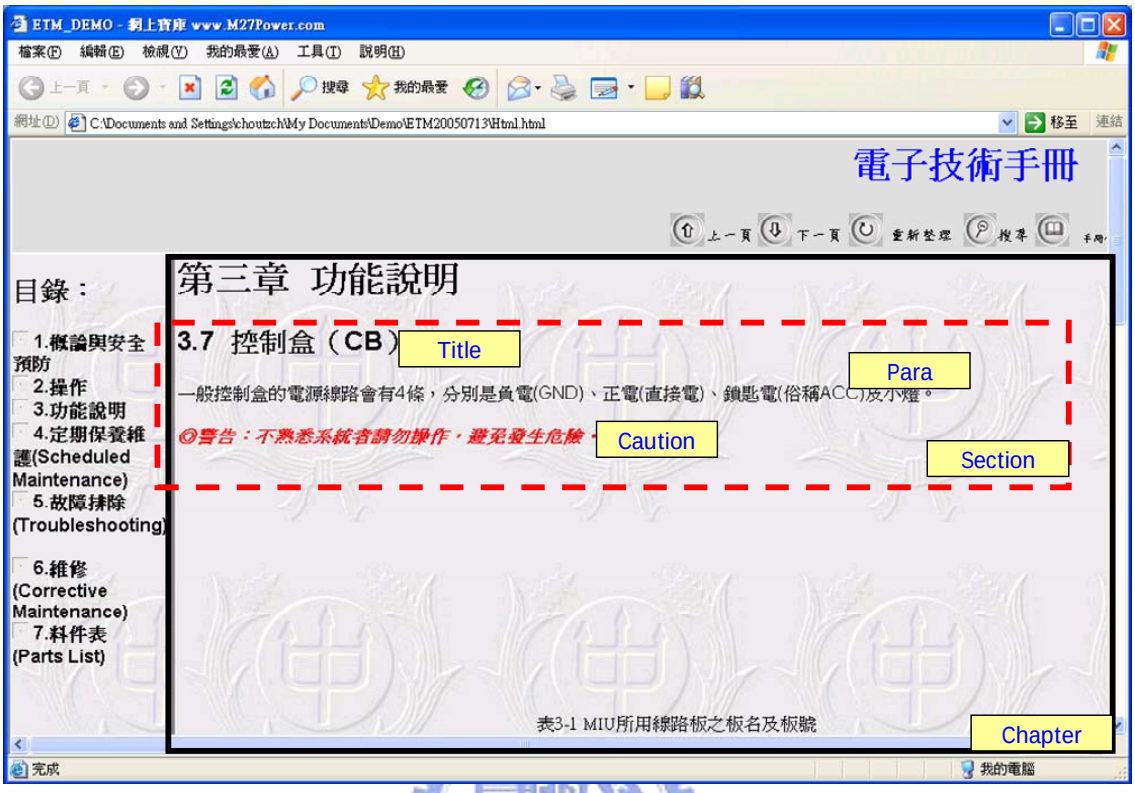


圖 4-2: ETM 範例

```
<DIV R-Pattern=" ((0|1|2)^*)" >
    <h3>
        <FONT face=Arial size=5>3.7  控制盒（CB）</FONT>
    </h3>
    <p>
        <FONT face=Arial>一般控制合的電源線路會有四條……</FONT>
    </p>
    <h4>
        <I><FONT color=#ff0000>◎警告： ……</FONT></I>
    </h4>
</div>
```

表格 4-1: ETM sample 網頁的部分 HTML

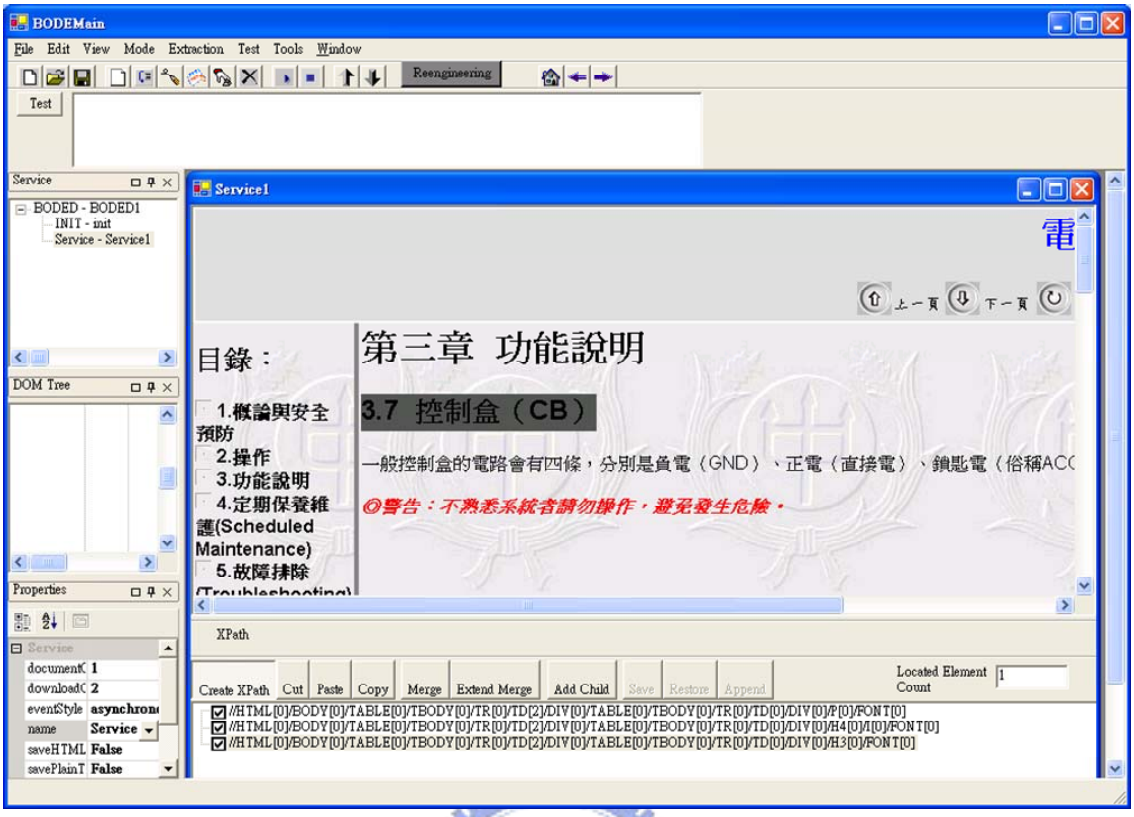


圖 4-3: 用 BODE 萃取 ETM 網頁

圖 4-3 是用 BODE 系統來萃取這個 sample 網頁，而產生出相對應的 BODE Script 如表格 4-2:

```
<BODED name="BODED1" ·····················>
  <INIT service="Service1" url="···········" />
  <Service to="Chapter" ·················>
    <Foreach Name="Foreach1" from=""
xpath="//HTML[0]/BODY[0]/TABLE[0]/TBODY[0]/TR[0]/TD[2]/DIV[0]/TABLE[0]/TBODY[0]/TR[
0]/TD[0]/DIV[position()&gt;=0]" to="Section">
      <VAR name="Var1" xpath="/H3[0]" bindtoforeach="False" to="Title"/>
      <VAR name="Var2" xpath="/p[0]" bindtoforeach="False" to="Para" />
      <VAR name="Var3" xpath="/h4[0]" bindtoforeach="False" to="Caution"/>
    </Foreach>
  </Service>
</BODED>
```

表格 4-2: 用 BODE 萃取 ETM sample 網頁產生的 BODE Script

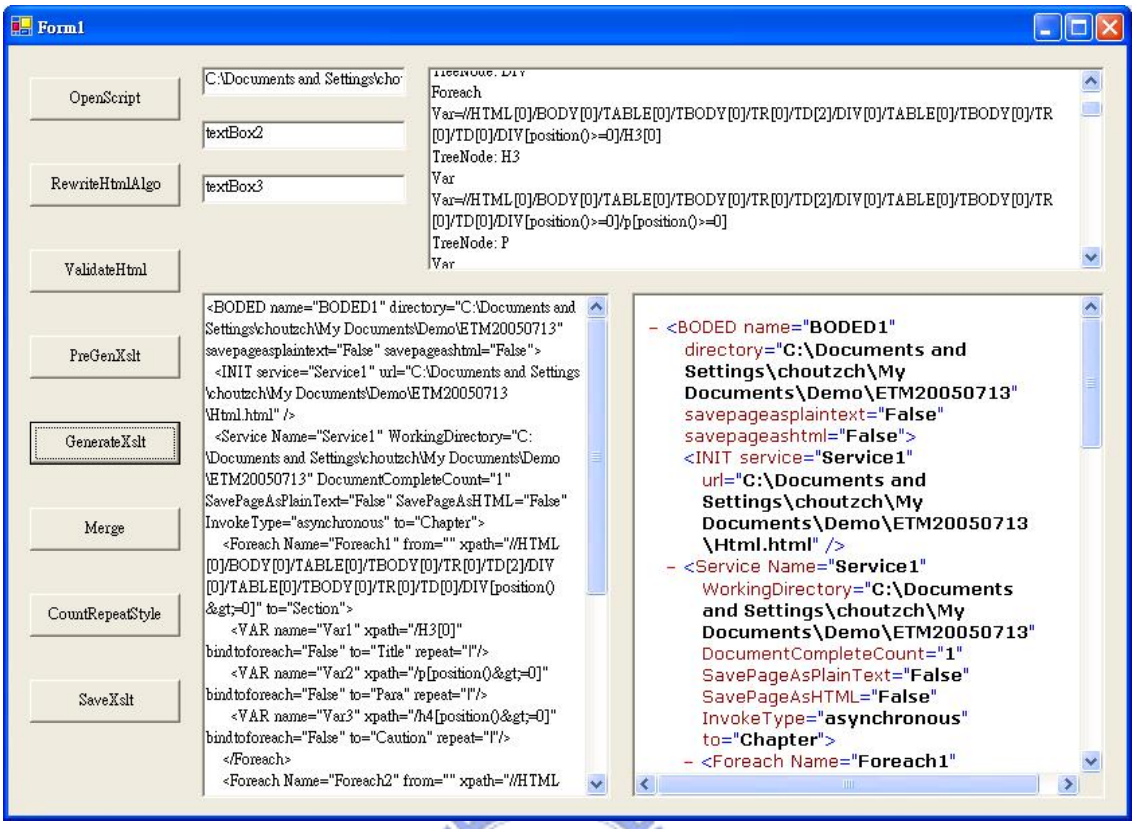


圖 4-4: XSLT Generator

圖 4-4 是我們的 XSLT Generator，必須輸入 BODE Script 及 sample 網頁，經過表格 3-1 的演算法計算後，最後會產生出相對應的 XSLT。

圖 4-5 是利用產生出來的 XSLT 套上 XML 資料檔，最後呈現出來的結果，將依照 sample 網頁的特性來做呈現，Chapter 下有許多的 Section，而每個 Section 下的資料則照資料順序做呈現。

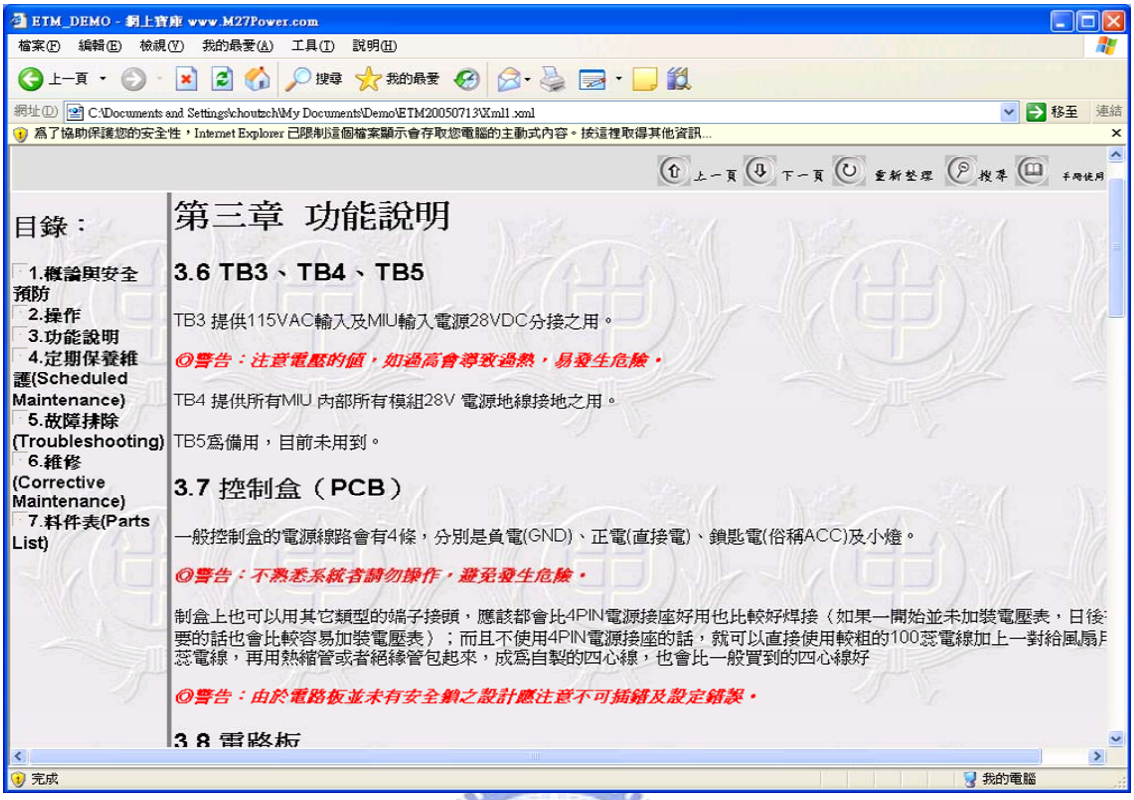


圖 4-5: 用產生的 XSLT 來轉換 XML Document

4.2.2 應用實例二

我們第二個 Demo 的例子是將此網頁再生系統用於整合三間書店網頁的書籍資料上，利用此網頁再生系統可以得到三種不同的 XSLT，當資料在資料庫中整合後，我們可以套用任一種 XSLT 來產生不同樣式的網頁，圖 4-6、圖 4-7、圖 4-8 分別是三個書店網頁的樣式。

書店一的書籍資料有 Title、Author、Price、Year、ISBN，且畫面的樣式是以一行標題列加上紅色、綠色、黃色的樣式來做呈現，如圖 4-6。

書店二的書籍資料有書名、作者、價錢，且畫面的樣式是一行標題列後用白色綠色的順序來呈現，如圖 4-7。

書店三的書籍資料有 Title、Price、OnSalePrice、Author，且畫面的樣式是 Price 這行用黃色，OnSalePrice 那行用粉紅色，如圖 4-8。

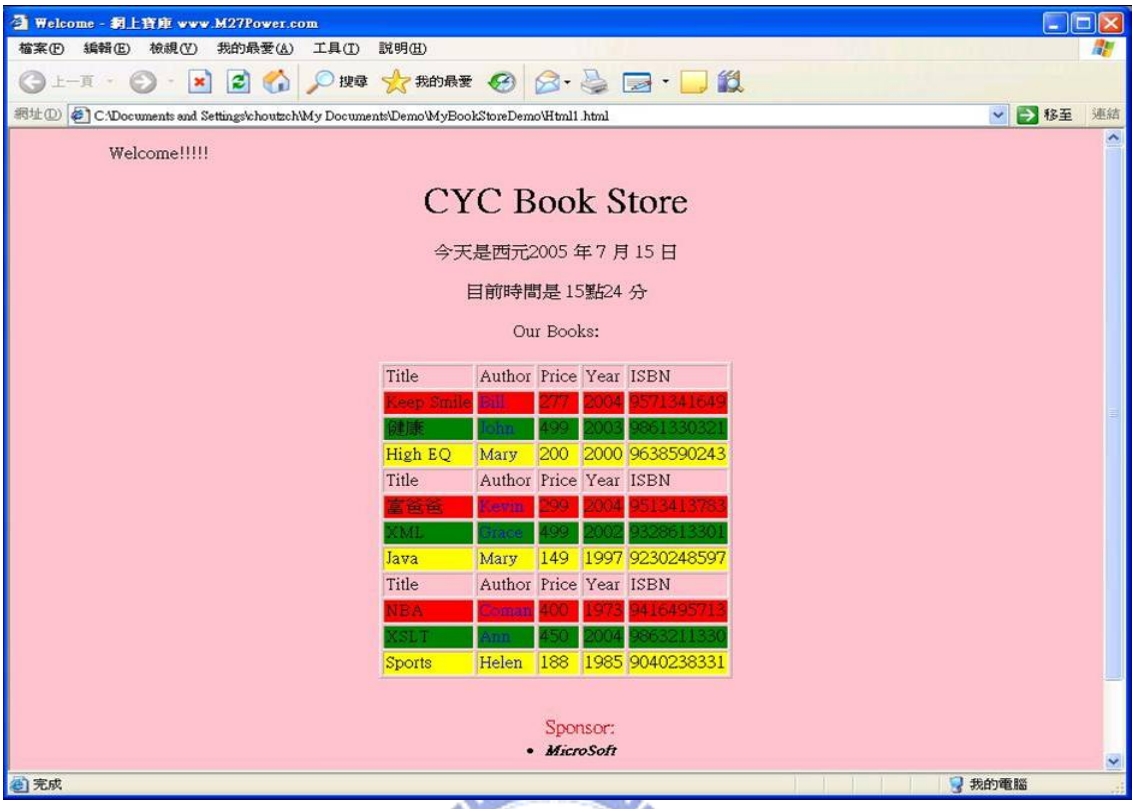


圖 4-6: 書店一的網頁樣式

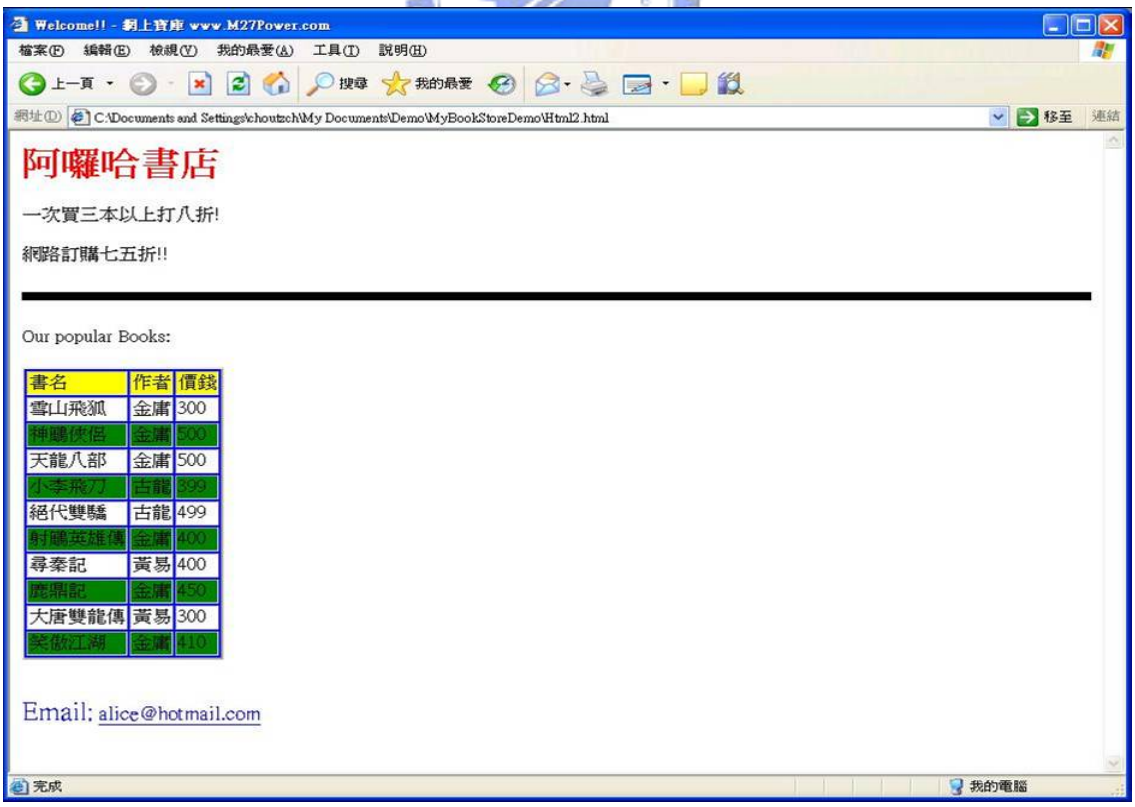


圖 4-7: 書店二的網頁樣式

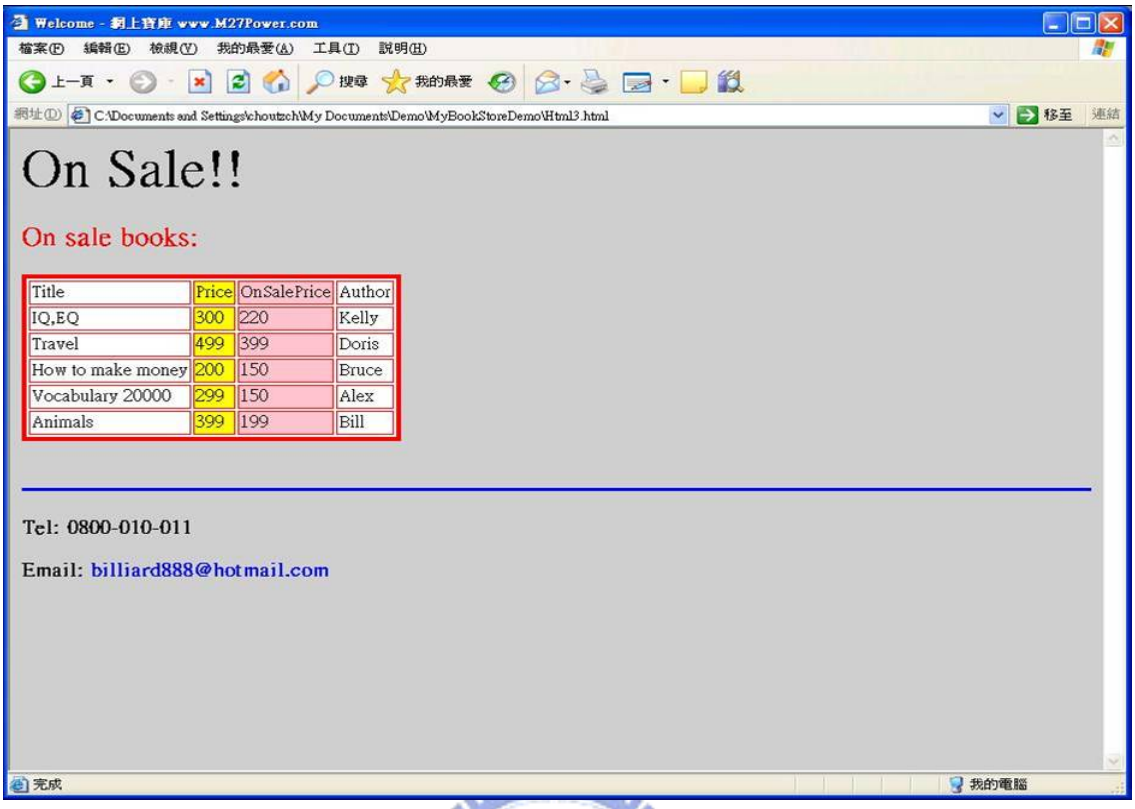


圖 4-8: 書店三的網頁樣式

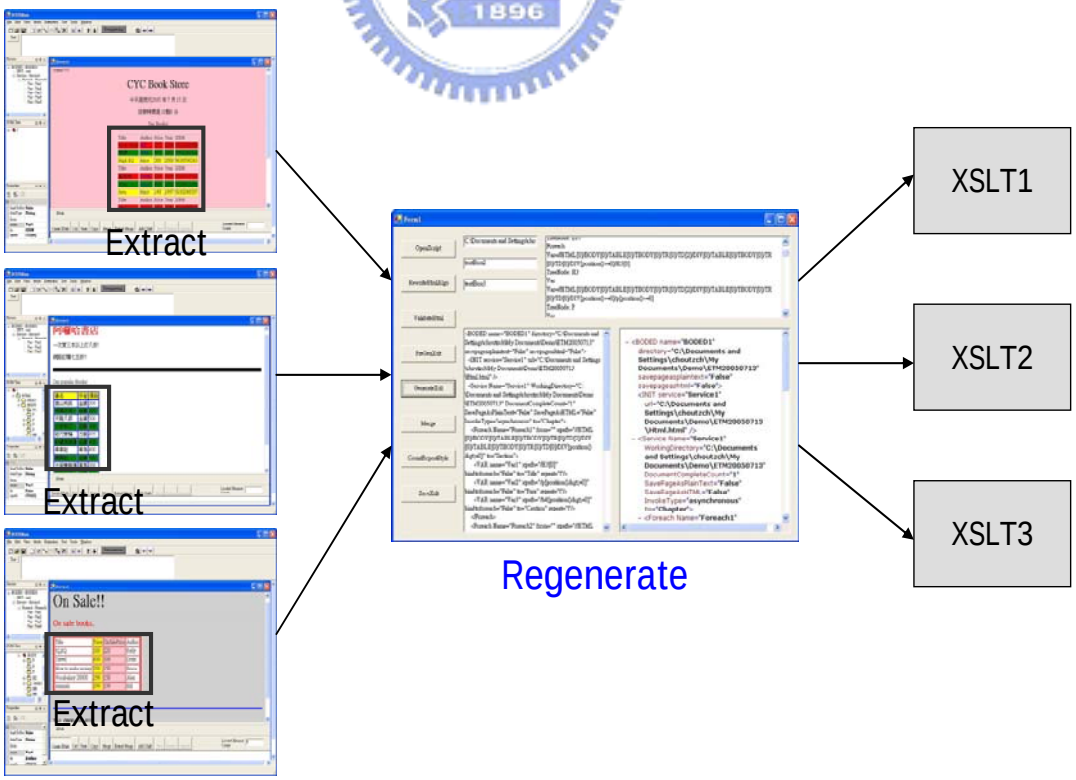


圖 4-9: 針對三個 sample 去產生各自的 XSLT

接著利用 BODE 系統分別萃取這三個 sample 網頁的書籍資料存到 XML 資料庫文件中，於是此 XML 資料庫文件中有這三間書店的書籍資料，如表格 4-3，再利用我們的網頁再生系統產生相對應的三個 XSLT 檔案，如圖 4-9。

```
<!--書店一的資料-->
<Book>
     <Title>Keep Smile</Title>
     <Author>Bill</Author>
     <Price>277</Price>
     <Year>2004</Year>
     <ISBN>9571341649</ISBN>
</Book>
.....

<!--書店二的資料-->
<Book>
     <Title>鹿鼎記</Title>
     <Author>金庸</Author>
     <Price>450</Price>
</Book>
.....

<!--書店三的資料-->
<Book>
     <Title>IQ,EQ</Title>
     <OnSalePrice>220</OnSalePrice>
     <Price>300</Price>
     <Author>Kelly</Author>
</Book>
.....
```

表格 4-3: 萃取三間書店書籍資料後的 XML 資料檔

最後就可以利用產生的這三個 XSLT 去再生網頁，呈現出剛剛的萃取的三間書店書籍資料，當然有些欄位會空著，因為每間書店的書籍格式不同，如圖 4-10、圖 4-11、圖 4-12。

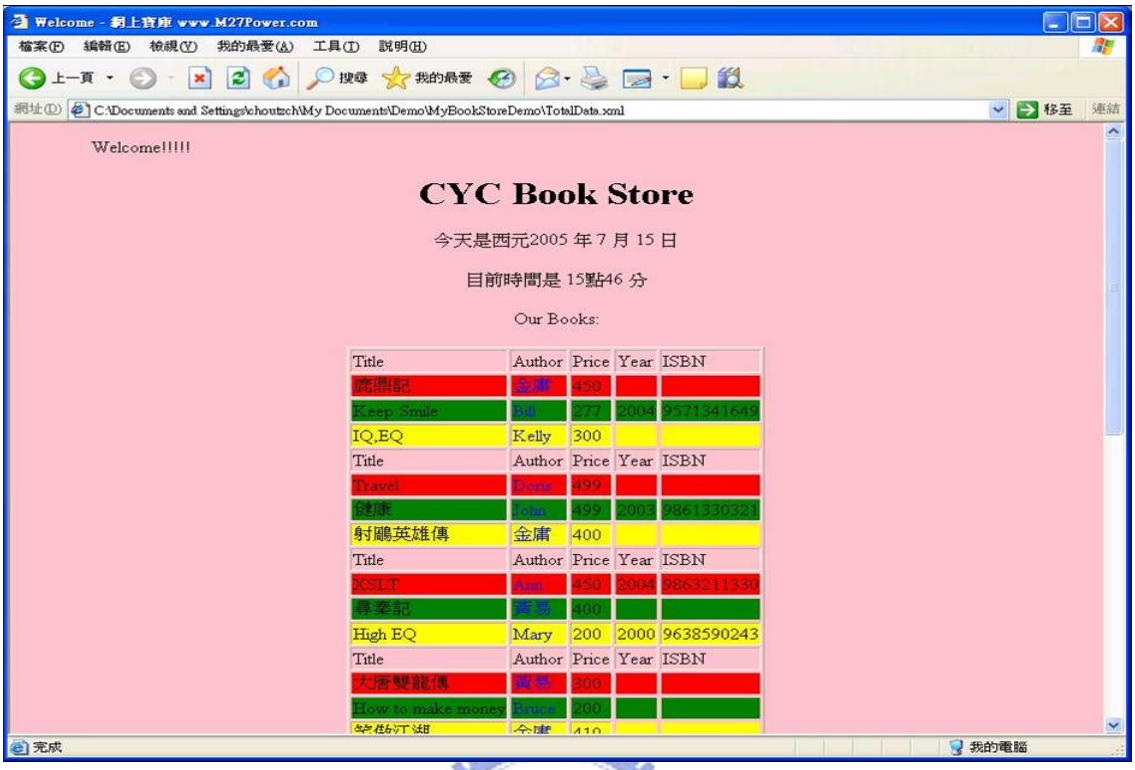


圖 4-10: 用產生的 XSLT1 來套資料結果

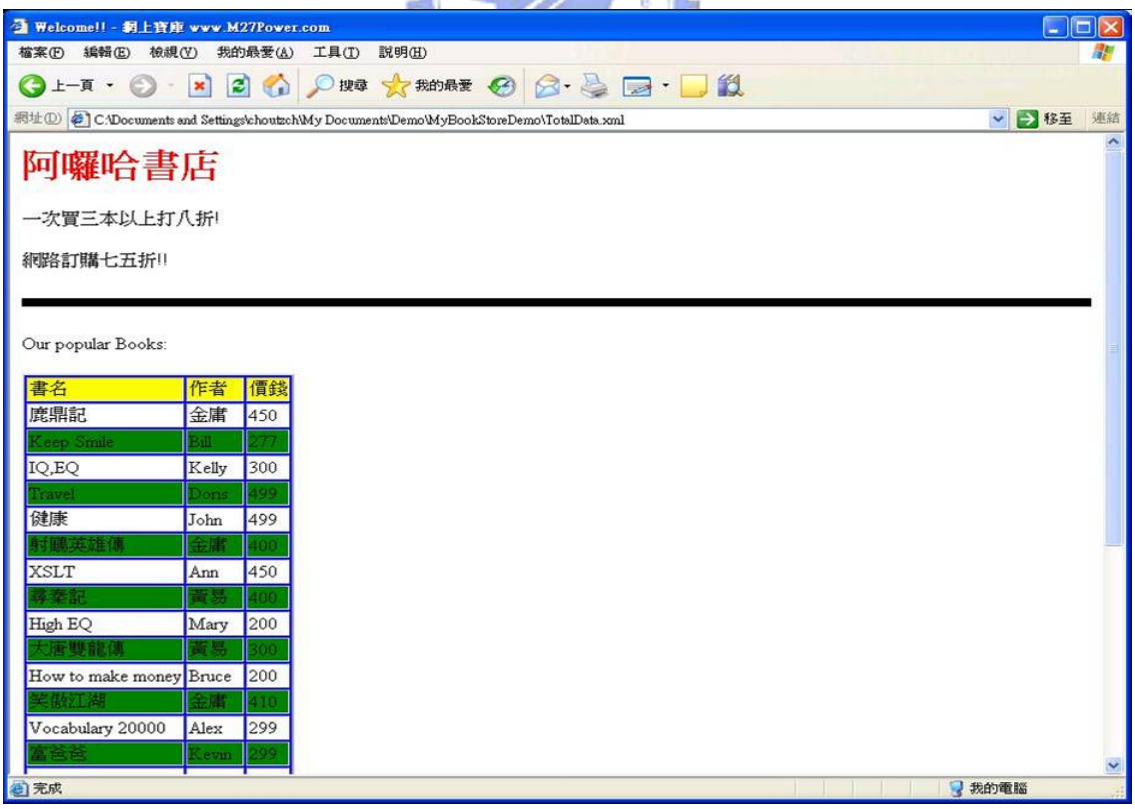


圖 4-11: 用產生的 XSLT2 來套資料結果

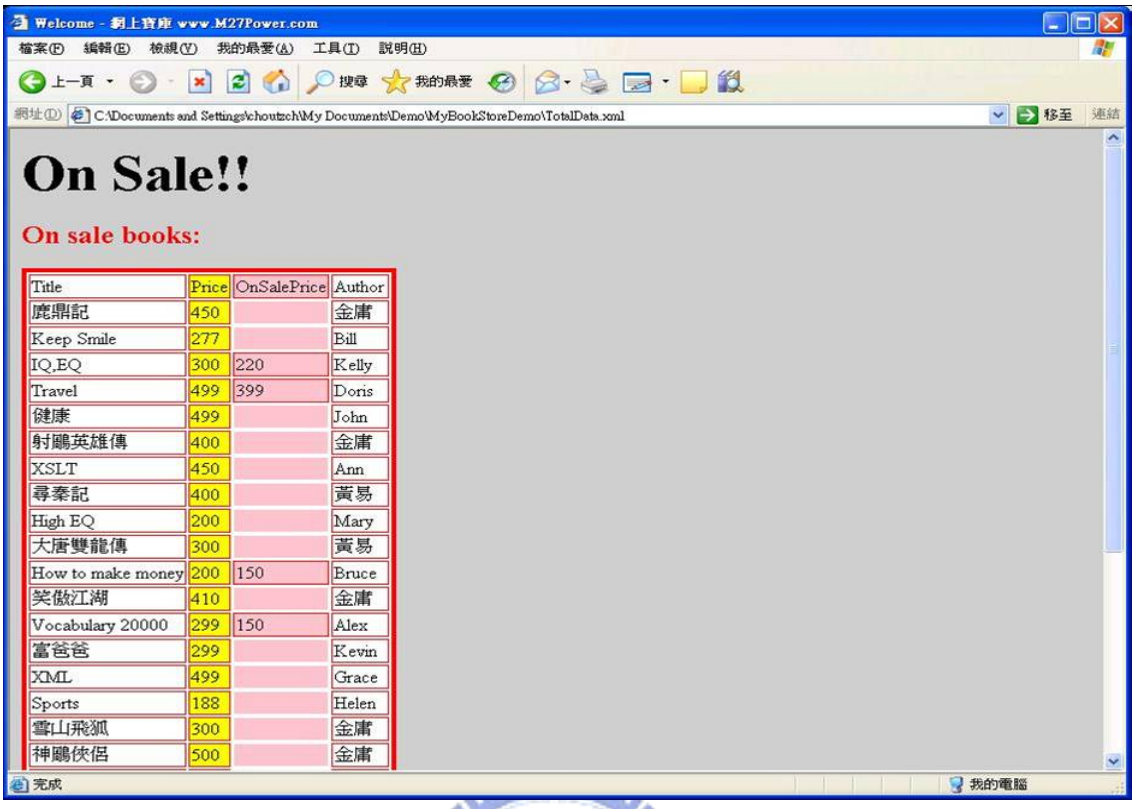


圖 4-12: 用產生的 XSLT3 來套資料結果

第五章 結論與未來工作

5.1 結論

本論文研究 XML 文件的資料擷取和資料呈現二個功能，並建立自動化的資料擷取與呈現系統。在資料擷取方面，本論文利用視覺化工具 BODE 系統來達成，而在資料呈現方面則以 XSLT 來達到相關功能。

開發 XSLT 是一項與輸出端有高度關聯性的程序，需要美工設計人員與程式設計師緊密的互動，因此撰寫 XSLT 是耗費時間的一項工作，而本論文的目的是研究 XSLT 的自動生成。本論文的主要概念是以 BODE 系統為工具，先製作一個 sample 網頁並對其進行資料擷取，藉由紀錄資料擷取的 BODE Script，我們的工具將會學習來源文件（也就是 sample 文件 S）與目的文件（也就是 XML 資料庫文件）之間的映對關係，進而計算出 XSLT，達到自動化網頁再生的目的。

計算 XSLT 的做法主要是使用 regeneration tree，它基本上是 sample 文件 S ，再加上一些屬性來描述文件 S 如何擴展成結果網頁。Regeneration tree 是由 BODE Script 與 sample 文件 S 兩者計算而得。本論文設計並實作可以將 regeneration tree 生成 XSLT 的演算法。

5.2 未來工作

本篇論文所提出之演算法所研究之主題有許多相關的研究課題，我們在這裡提出以下可以擴充的課題:

- 第一點，由於 XSLT 語法是描述 Style Sheet 的語法，而非一般程式語言，因此在功能方面有許多限制，本演算法可以考慮除了 XSLT 以外，再加上其他程式語法，使自動化再生更具效率。

- 第二點，我們的演算法所產生的 XSLT 文件 *XS* 是直接讀取 XML 資料庫文件來輸出網頁。我們建議在 *XS* 與 XML 資料庫文件之間加入 ViewLet 模組，這個 ViewLet 的功能是讀取 XML 文件的資料，並處理關聯（Relation）或是資料排序過濾等，ViewLet

也可以讀取 XML 格式以外的資料檔案，例如關聯式資料庫、格式化文字檔、半結構化電子文件等，如此能使大幅擴充本演算法所能處理的文件的範圍，如圖 5-1。

- 最後，本論文所提出的演算法所處理的來源文件與目的文件分別是 HTML 與 XML，這部份可以擴充為兩者皆為 XML L 文件，使其更具一般性。

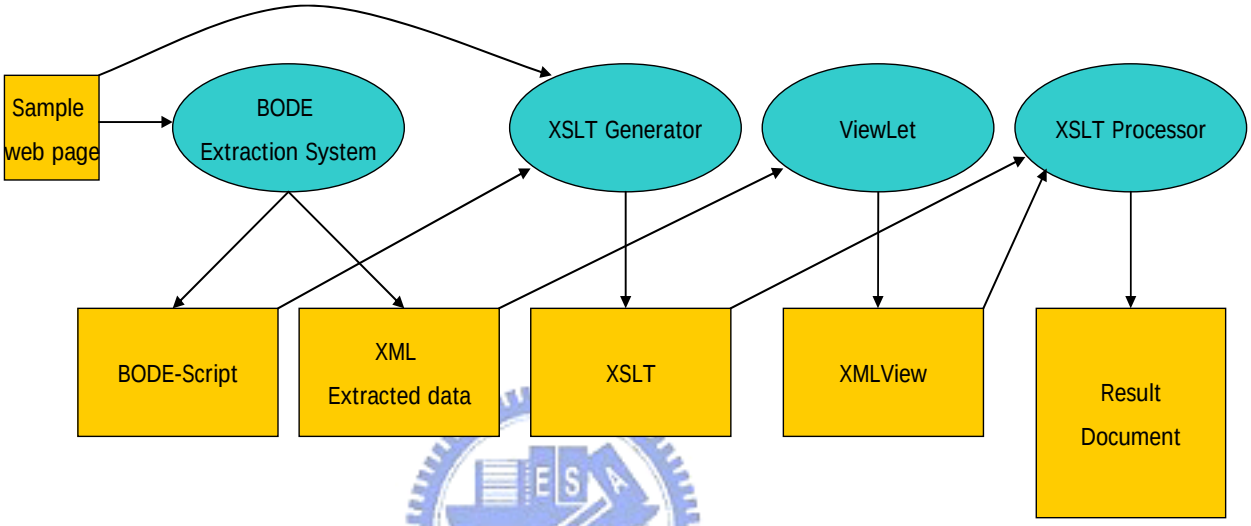


圖 5-1: 加入 ViewLet 的網頁再生流程圖

參考資料

[1] http://www.w3c.org.

[2] HTML-Kit. http://www.chami.com/html-kit/.

[3] TIDY. http://www.w3.org/People/Raggett/tidy/.

[4] Xml spy, 2001. http://www.xmlspy.com.

[5] http://tw.yahoo.com.

[6] I-Chen Wu, Jui-Yuan Su, Loon-Been Chen, W. C. Chien, and C. T. Lee, “DESDL: A Data Extraction Service Description Language”, In the Proceedings of the International Computer Symposium, NDHU, Hua-lien, Taiwan, December 2002.

[7] “Performance Specification MIL-PRF-87269A Data Base, Revisable-Interactive Electronic Technical Manuals, for the support of ”, Tri-Service Working Group for Interactive Electronic Technical Manuals, October 1995.

[8] James E. Giles III (1994), “Interactive Electronic Technical Manuals (IETMs), Part 2,”CALS/Enterprise Integration Journal, Fall 1994, p72.

[9] “Military Specification MIL-M-87268 Manuals, Interactive Electronic Technical: General Content, Style, Format, and User-Interaction Requirements, ”Tri-Service Working Group for Interactive Electronic Technical Manuals, November 1992。

[10] I-Chen Wu, Jui-Yuan Su, and Loon-Been Chen, “Browser-Oriented Data Extraction”, in 2004 International Computer Symposium (ICS2004), Taipei, December 2004.

[11] E. Pietriga and J.-Y. Vion-Dury, “VXT: Visual XML Transformer”. IEEE Symposium on Visual/Multimedia Approaches to Programming and Software Engineering (Human Centric Computing Languages and Environments), September 2001.

[12] Benoît Marchal, “How an XSLT processor works”, in IBM developerWorks XML zone, March 2004.

[13] Eric Jorgensen and Joseph Fuller (1994), “Interactive Electronic Technical Manuals (IETMs), Part 1”, CALS/Enterprise Integration Journal, Summer 1994, p68

[14] I-Chen Wu, Jui-Yuan Su, and Loon-Been Chen, “A Web Data Extraction Description Language and Its Implementation”, The 29th Annual International Computer Software and Application Conference (COMPSAC 2005), Edinburgh, Scotland, July 2005.

[15] Zhenjiang Hu, Shin-Cheng Mu, Masato Takeichi, “A programmable editor for developing structured documents based on bidirectional transformations”. In Partial Evaluation and Semantics-Based Program Manipulation, pp. 178-189. August 2004.

[16] Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierc, and Alan Schmitt. “A language for bi-directional tree transformations”. Technical Report Technical Report MS-CIS-03-08, Department of Computer and Information Science University of Pennsylvania, August 2003.

[17] W3C Consortium. “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation, October 2000. http://www.w3c.org/TR/2000/REC-xml-20001006.

[18] W3C Consortium. “Hyper Text Markup Language”, January 1998. http://www.w3c.org/Markup.

[19] W3C Consortium, “HTML 4.01 Specification” W3C Recommendation, December 1999. http://www.w3c.org/TR/html4.

[20] W3C Consortium. “XML Path Language (XPath) 1.0”, W3C Recommendation, November 1999. http://www.w3c.org/TR/xpath.

[21] W3C Consortium. “XML Path Language (XPath) 2.0”, W3C Working Draft, April 2005. http://www.w3c.org/TR/xpath20.

[22] W3C Consortium. “XSL Transformations (XSLT) 1.0”, W3C Recommendation, November 1999. http://www.w3c.org/TR/xslt.

[23] W3C Consortium. “XSL Transformations (XSLT) 2.0”, W3C Working Draft, April 2005. http://www.w3c.org/TR/xslt20.

[24] P. Merrick and C. Allen, “Web Interface Definition Language (WIDL)”, http://www.w3c.org/TR/NOTE-widl-970922, September 1997.

[25] Mark G. Wales, “WIDL: Interface Definition for the Web”, IEEE Internet Computing, Vol. 3, No. 1, pp. 55-59, January 1999.