

DOM Level 3 的 Load 與 Save

By 蘇國鈞

monster@iii.org.tw

用過 DOM 的人都知道，DOM Level 2 本身並不支援讀寫 XML 文件的功能。雖然透過 Java API for XML Processing (JAXP) 的協助，目前有標準的作法可以讀取 XML 文件的內容，建立出 DOM Tree，但是並沒有標準的作法可以將 DOM Tree 寫成 XML 文件，也因此衍生出以下各式各樣的解決方式。

自行撰寫程式

因為 DOM Level 2 有標準的 API 可以新增、刪除、查詢、與修改 DOM Tree 的內容，所以您可以撰寫程式去「爬樹」，將所得結果自行輸出成 XML 文件。如果對這一類的作法有興趣的話，您可以參考 Apache 的 Crimson 與 Xerces2 等 XML Parser 的範例程式。

這種作法的優點是「跨 XML Parser」，完全不受限於底層的 XML Parser 實作產品。缺點則是「累死自己」，因為寫程式爬過樹的人都知道，爬樹並不是件輕鬆的事。

使用 XML Parser 的特異功能

Apache 的 Crimson 與 Xerces2 都提供一些「特殊」的類別，可以幫助您輕鬆地將 DOM Tree 輸出成 XML 文件。所謂特殊，就是「非標準」的意思。換句話說，您只要用了這項功能，您的程式碼某種程度就綁死在某個特定的 XML Parser 實作產品，不能夠再隨便更換其他的 XML Parser 了。

使用 TrAX 的標準功能

Transformation API for XML (TrAX) 是 JAXP 的一部份，用來搭配 XSL 文件進行

XSLT。進行轉換的時候如果沒有提供 XSL 文件的話，TrAX 就會執行所謂的 Identity Transform，也就是原封不動地將 DOM Tree 的內容轉換成您指定的輸出方式。好玩的地方也就在這裡：如果您將輸出的方式指定成檔案的話，不就可以將 DOM Tree 原封不動地轉換成 XML 文件嗎？

這種作法的優點也是「跨 XML Parser」，因為我們使用的是標準的 TrAX。缺點則是要「學習 TrAX」，不過這並不是什麼大問題，因為 TrAX 很簡單，一下子就可以學會，更何況它也是標準 API。

使用 DOM Level 3 的 Load and Save

2004 年四月，World Wide Web Consortium (W3C) 將 DOM Level 3 的 Core、Load and Save、與 Validation 等部分，對外公佈為 W3C 的 Recommendation。以 Java 來說，至少 Apache Xerces2 提供了 DOM Level 3 的實驗性實作產品。

因為 Xerces2 預設並沒有啟用 DOM Level 3 的支援，所以如果要使用 DOM Level 3 的 Load and Save 功能的話，要嘛自行編譯 Xerces2，否則請下載啟用 DOM Level 3 功能的特殊編譯版本，目前最新的是 beta2-dom3-Xerces-J-bin.2.6.2.zip。解開之後會得到四個 JAR 檔案，您只要想辦法擴充您的 CLASSPATH，讓 Java 程式先找到這個支援 DOM Level 3 的 XML Parser 就可以了。透過 LSParser 與 LSSerializer 等物件，您就可以在 XML 文件與 DOM Tree 之間往返自如。程式範例如下：

```
import org.w3c.dom.*;
import org.w3c.dom.bootstrap.*;
import org.w3c.dom.ls.*;

public class MyDOM3
{
    public static void main(String[] args)
    {
        try
        {
            System.setProperty(DOMImplementationRegistry.PROPERTY,
                "org.apache.xerces.dom.DOMImplementationSourceImpl");
            DOMImplementationRegistry registry = DOMImplementationRegistry
                .newInstance();

            DOMImplementation dom = registry.getDOMImplementation("LS");

            DOMImplementationLS dom3ls = (DOMImplementationLS) dom;
            LSParser parser = dom3ls.createLSParser(
                DOMImplementationLS.MODE_SYNCHRONOUS, null);

            Document doc = parser.parseURI("Source.xml");
```

```
        LSSerializer serializer = dom3ls.createLSSerializer();
        serializer.writeToURL(doc, "Result.xml");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
}
```

孰優孰劣？

這個問題見仁見智，端視應用場合而定。自行撰寫程式客製化的程度最高，但是也是最辛苦；使用 XML Parser 特異功能可能是最輕鬆的方式，但是就必須使用特定的 XML Parser。TrAX 與 DOM Level 3 的 Load and Save 都算是標準 API，但是目前 XML Parser 對 DOM Level 3 的支援程度還不算太好，甚至沒有支援，以現階段來說，使用 DOM Level 3 跟綁死特定 XML Parser 其實沒有太大差別。

這麼分析起來，使用 TrAX 似乎是目前比較可以接受的方式囉？

或許吧？

除了上面這些處理 DOM Tree 的方式之外，你也可以考慮改用其他的 API 來處理 XML 文件，比方說 JDOM 或 DOM4J。或是甚至用物件導向的方式來處理 XML 文件，比方說官方標準的 JAXB 或 Open Source 的 Castor。

處理一個問題的方式有很多種，各有巧妙不同，沒有絕對的好，也沒有絕對的壞，就看您從那個角度來看這個問題罷了！