XML動態網頁技術
XML Dynamic Page

國立台灣師範大學
資訊工程系
葉耀明

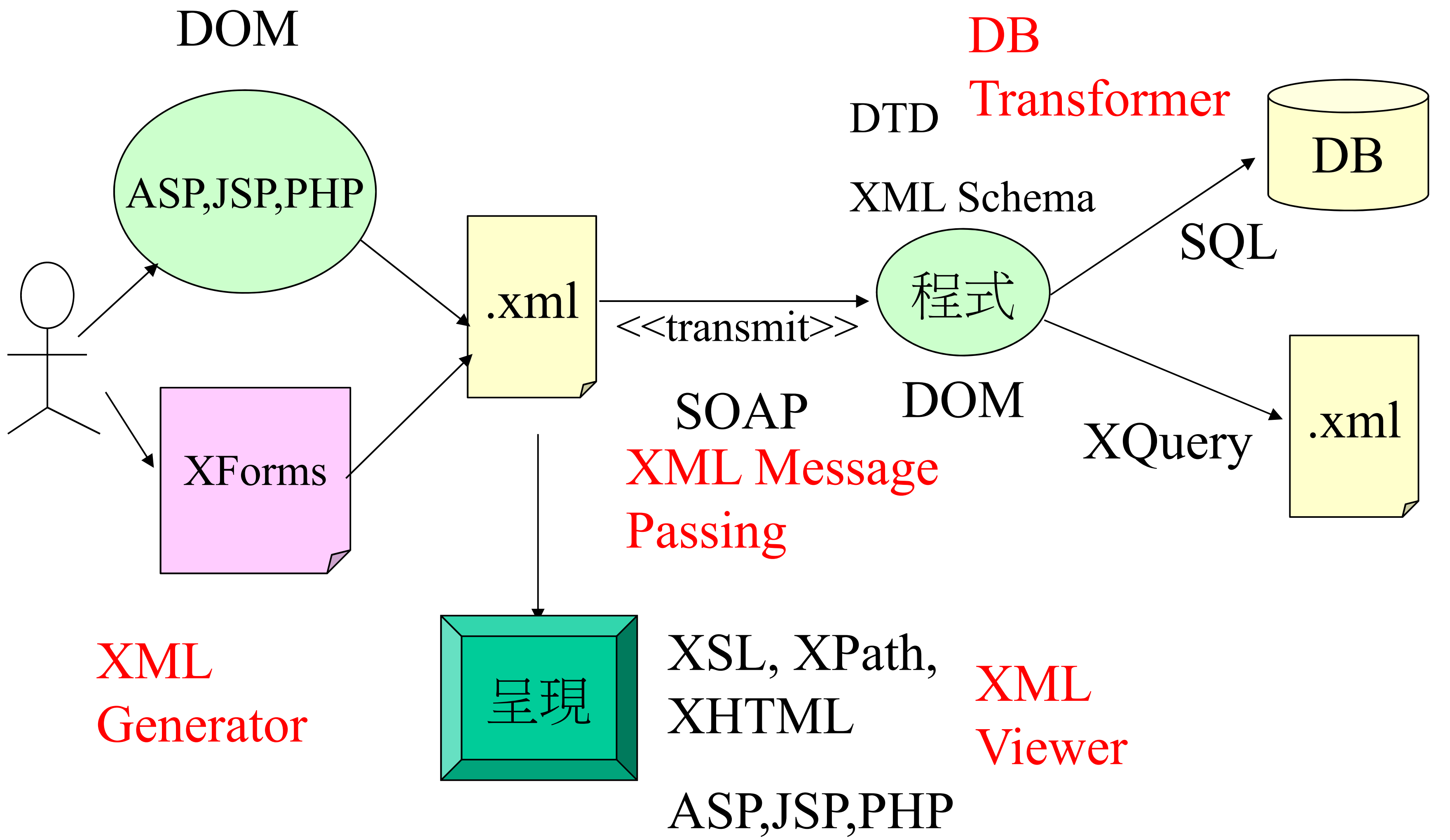
內容大綱

- Parser的概念(concept)
- DOM v.s. Data Base
- 動態網頁Dynamic Page Design Patterns
- DOM API
- 範例：One-Page Web Application
- XQuery：未來的動態網頁技術
- XForms
- 總結

XML動態網頁技術 (XML Dynamic Page Knowhow)

- XML文件的程式產生(XML document generation)
- DOM: W3C DOM, Java DOM, .NET DOM
- XForms
- DTD
- XML Schema
- XQuery
- DB Transform

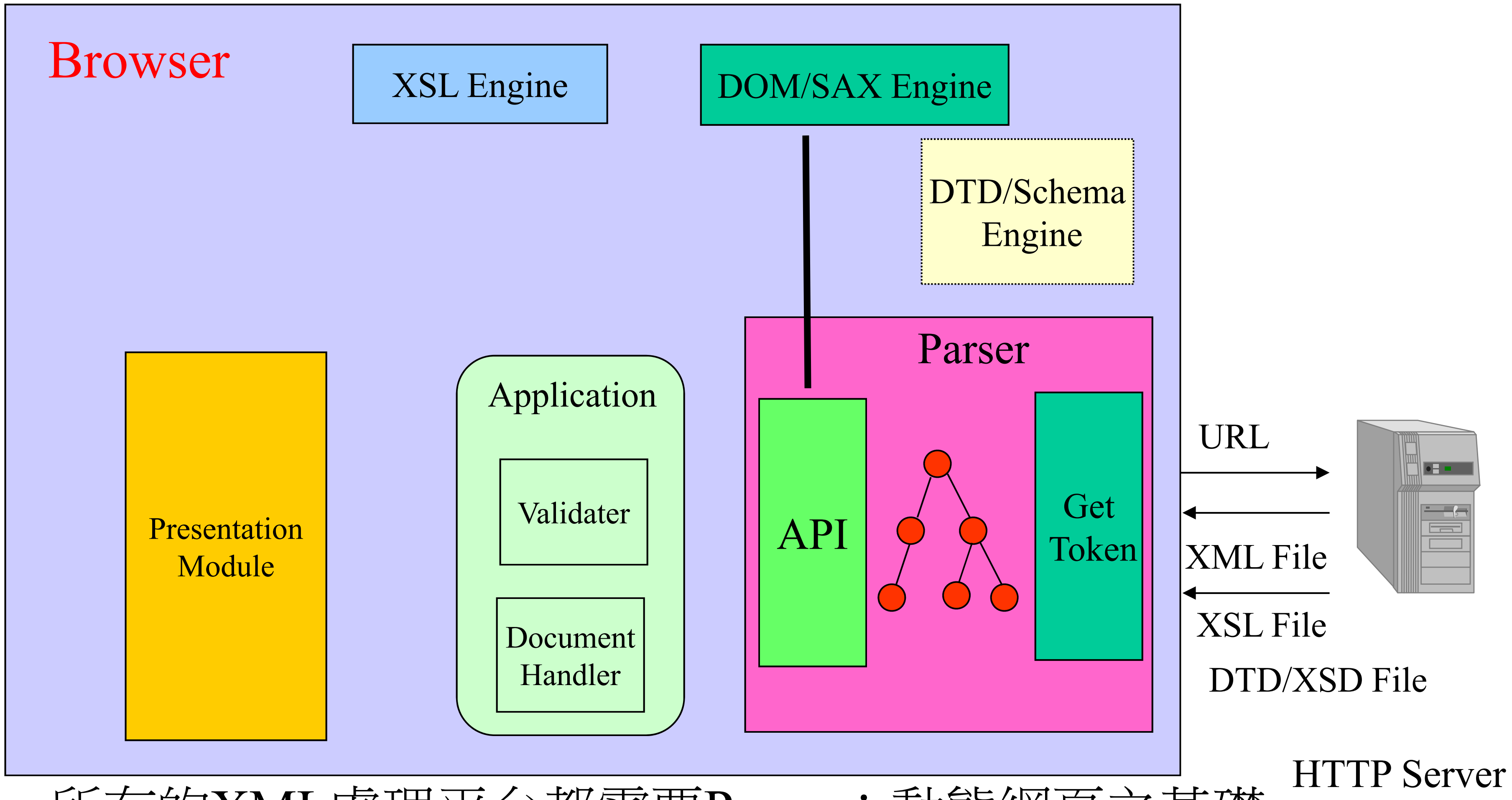
XML動態網頁技術運作情境
XML Dynamic Page Mechanism



四種方式對XML文件Query
(Four Types of Query for XML Document)

- DOM API:最低階(lowest level)
- XPath+XSLT
- XFilter
- XQuery: 最高階(highest level)

Inside Browser



所有的XML處理平台都需要Parser：動態網頁之基礎

(All XML platform need parser to handle XML document(foundation of XML processing)

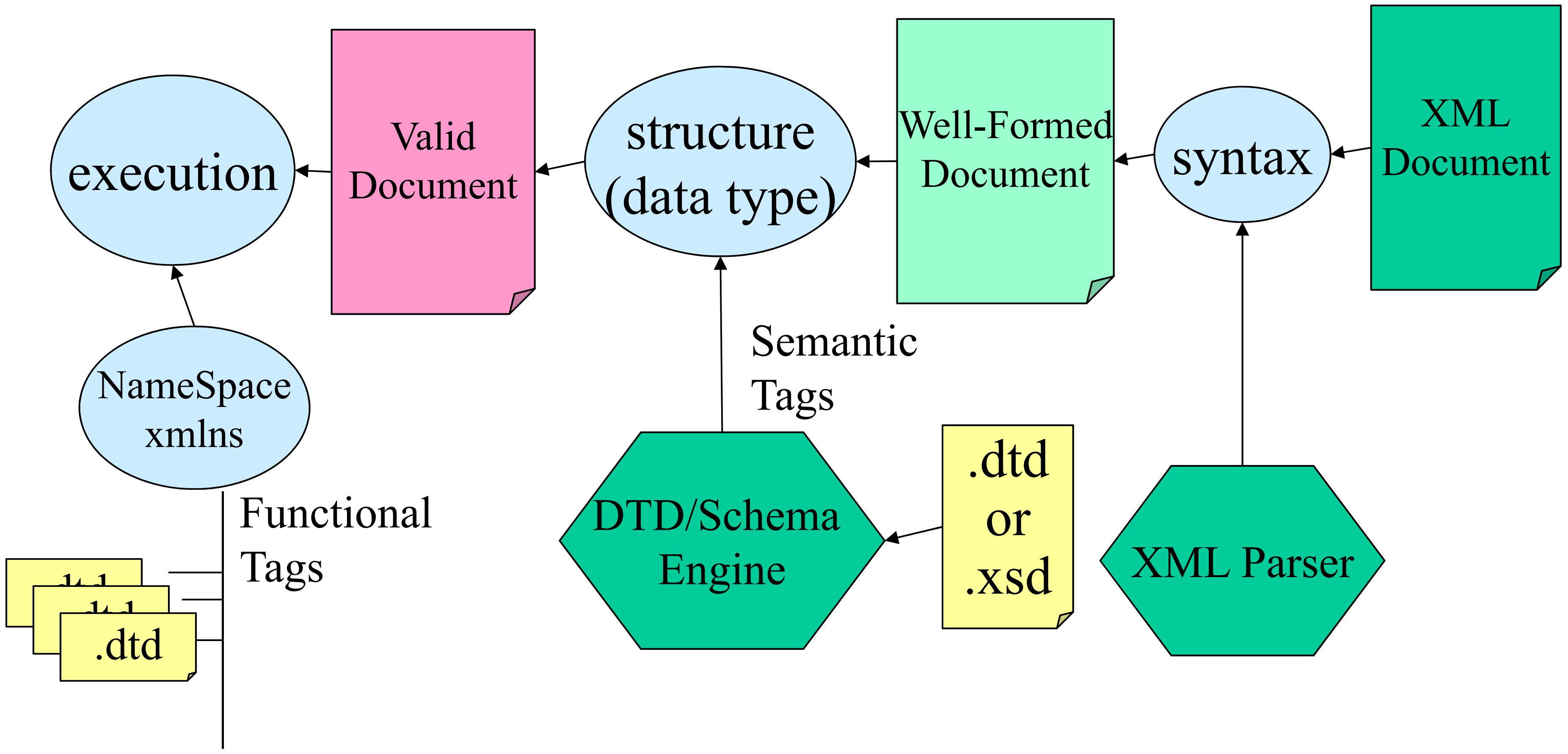
Parsers: (XML processor)

- 驗證XML文件為Well-formed Documents
- 驗證所有資料物件符合XML語法(XML Syntax validation)
  - 文件語法符合XML語法的規定(syntax conforms to the XML specification)
  - 元素形成一個單根節點的樹(elements form a hierarchical tree, with a single root node)
  - 除了所提供的DTD外，沒有參考到外部實體 (there are no references to external entities, unless a DTD is provided)

Parser的類型 (Types of Parser)

- non-validating: Parser只確保文件well-formed (only check well-formed)
- validating: Parser使用DTD來驗證well-formed資料的格式和內容(use DTD to validate well-formed document)
  - 文件樹和DTD定義樹的比對(matching between XML tree and DTD declared trees)
- Parser Implementation:
  - **Tree-based Parser: DOM Parser (Standard)**
  - **Event-driven Parser: SAX Parser (non-standard)**

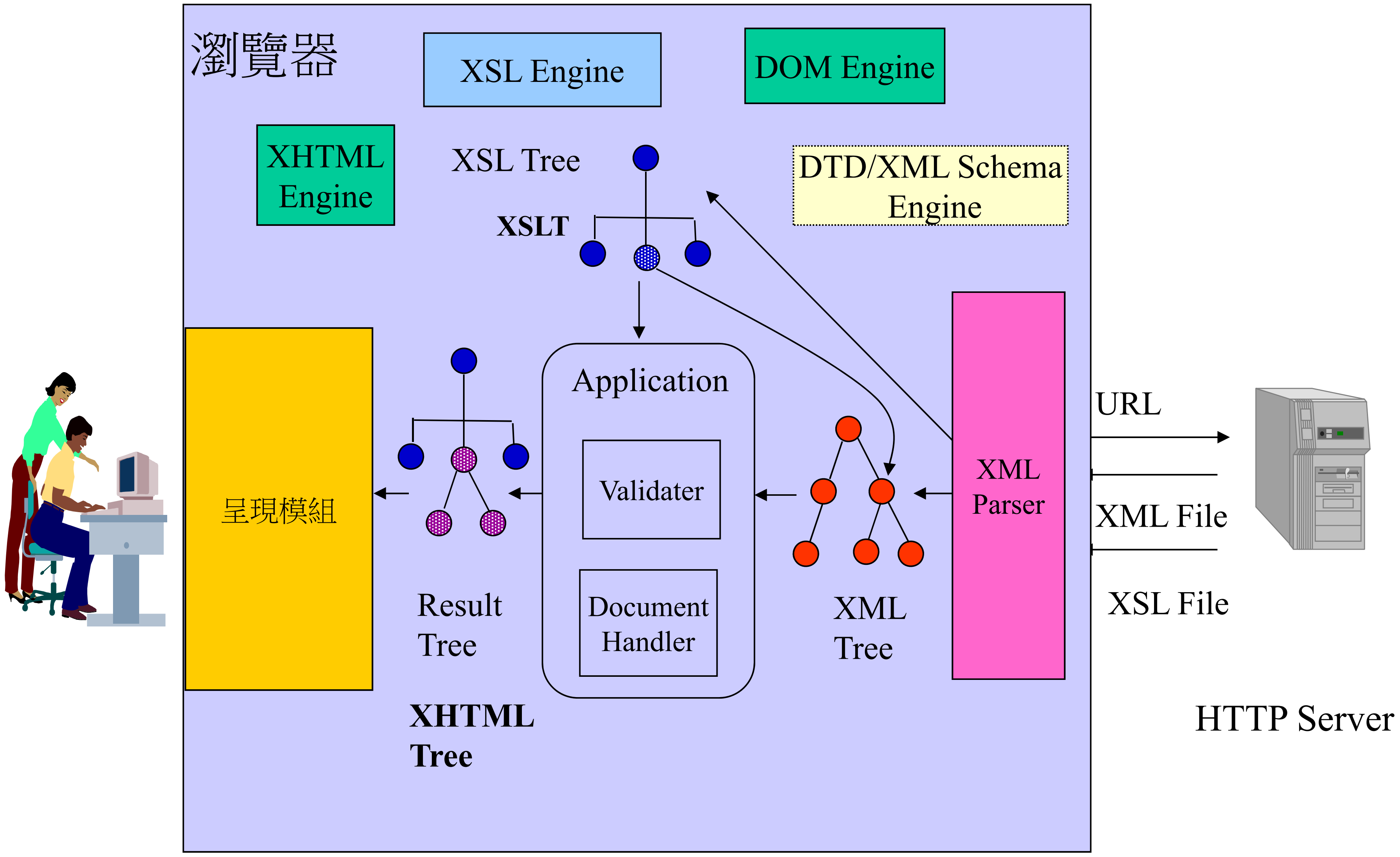
Parse, Validate, Execute



Tree-based Parser: DOM Parser

- 將XML文件轉換為DOM Tree (Transform XML document into DOM tree)
  - The DOM is a platform- and language-neutral interface that allows manipulation of tree-structured documents.
  - 為W3C標準(W3C standard)
  - Tree Traversal: random access
- Examples: Browser
  - MSXML by Microsoft (included in IE 5.0), XJParser, Office 2000 use XML for data exchange format
  - Mozilla project (by Netscape): “SeaMonkey” HyBrick by Fujitsu Lab. (SGML/XML browser)
  - FireFox

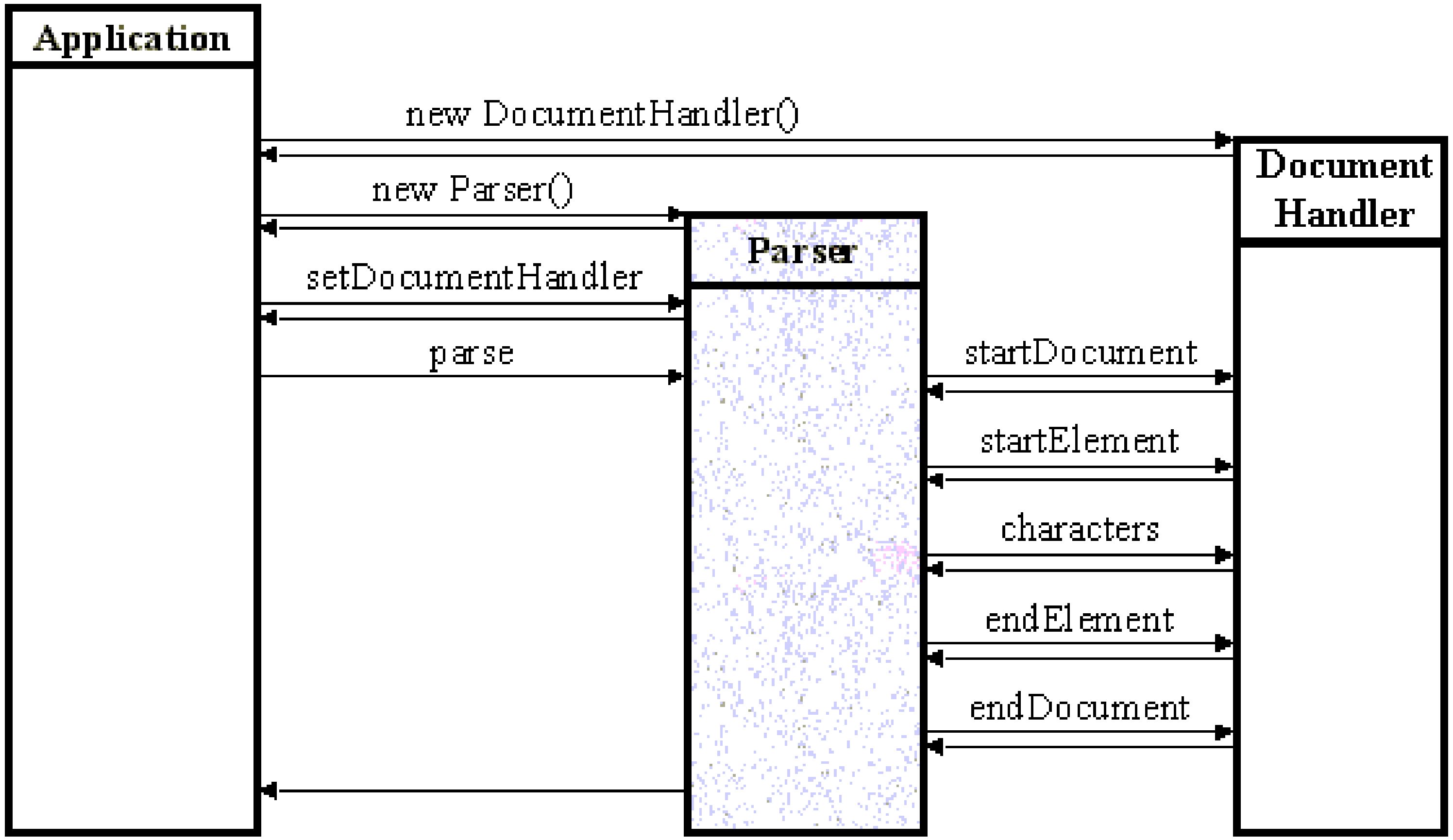
Tree-based Parser



Event-driven Parser: SAX Parser

- 利用event來做文件元素內容的溝通(Use event to provide the document functions)
  - 操作方式像 GUI and OS的API呼叫 (traditional execution of program code)
  - 並非W3C的標準(non-standard)
  - Sequential Traversal
- Examples:Browser外(non-browser platform)
  - expat by James Clark (in ANSI C), expatpp (in C++), XML::Parser (in Perl), Pyexpat(in Python), Java
- SAX: Simple API for XML

The Structure of SAX



比較Comparison Tree-based/Event-driven Parsers

- Tree-based Parser
  - 官方標準(主流)(standard)
  - 適合XML Datagram(檔案)(For XML datagram(file)
  - 佔用大記憶體空間，適合使用XML小檔案(Need large memory space, good for small file)
  - 剛發展時以Browser內為主(Browser platform)
- Event-driven Parser
  - 非官方標準(non-standard)
  - 適合XML Datagram(檔案)和Datastream(資料流)(good for both datagram and datastream)
  - 可使用XML大檔案(good for large files)
  - 剛發展時以Browser外為主(non-browser platform)

DOM(Document Object Model)

- 文件物件模型(Document Object Model)
  - A platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style and of a document
  - 包括XML文件的資料結構(DOM Tree)、處理DOM tree的API (include DOM tree and API)
- XML文件(Text格式)的統一內部資料結構：DOM Tree (Binary格式)
- XML document (text format) – DOM tree (binary format)

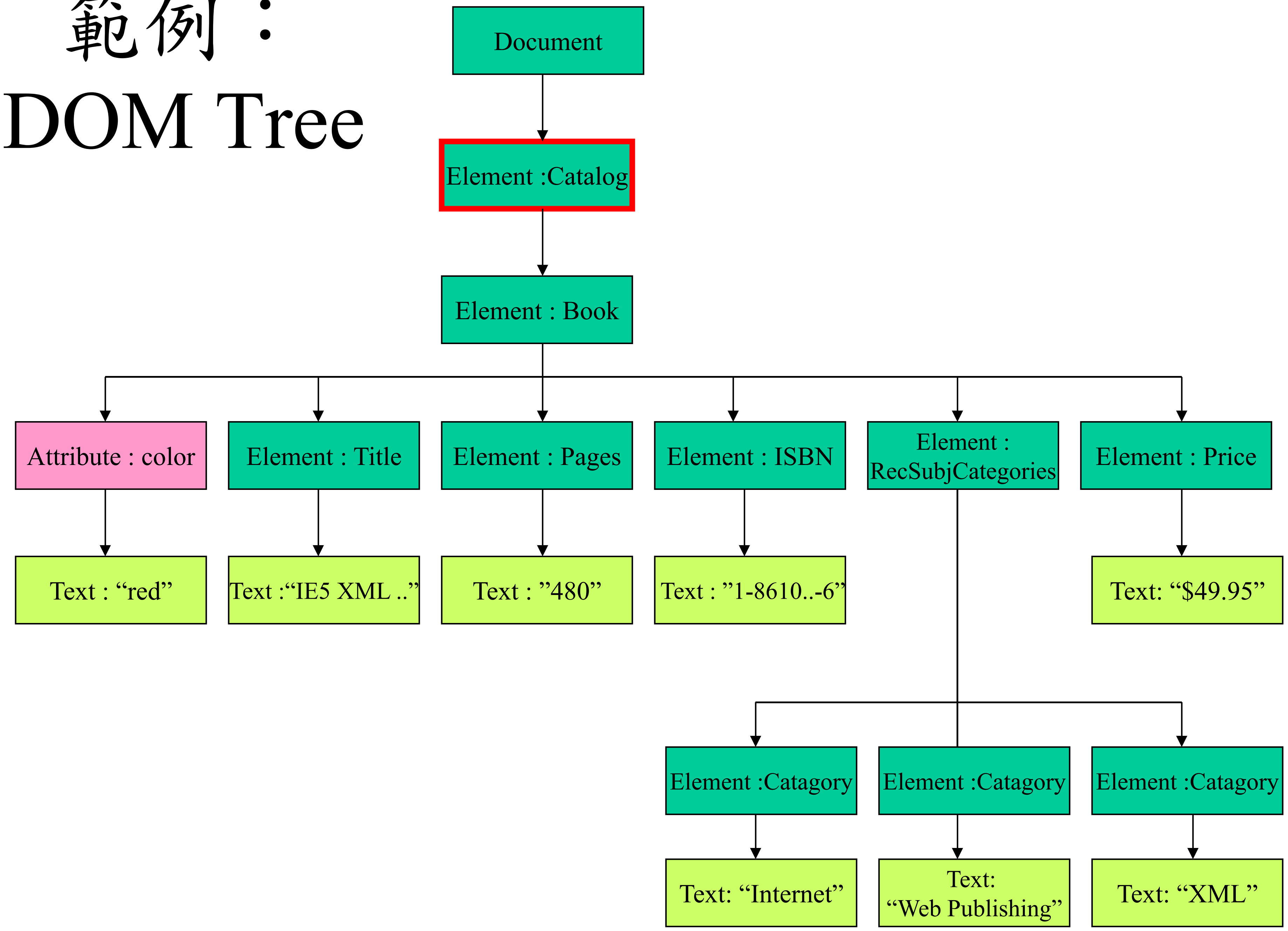
DOM Level

- Level 0: (Recommendation)併進Level 1
  - HTML-specific extension (定義HTML文件在DOM的處理)
- Level 1: (Recommendation)：datagram 1998/10/01 (2000/9/29 2nd Edition)
  - HTML-specific extension (定義HTML文件在DOM的處理)
  - API to access XML文件的內容(不包括DTD, Style sheet)
- Level 2: (Recommendation)：datastream
  - Core(2000/11/13), Views(2000/11/13), Events(2000/11/13), Style(2000/11/13), Traversal and Range(2000/11/13), HTML(2003/1/9)
  - Support for Namespace，Style sheet
  - Filtering: 過濾content
  - Event model
  - Range: 處理長文件(for large File)
- Level 3: (Recommendation)：data type 2004/4/7
  - Core(2004/4/7 Rec), Load and Save(2004/4/7 Rec), Validation(2003/10/15 Rec), Events(2003/3/31), XPath(2003/3/31)
  - Content Model and Load and Save
  - XPath

範例：XML文件

```
<Catalog>
   <Book color=“red”>
      <Title>IE5 XML Programmer’s Reference</Title>
      <Pages>481</Pages>
      <ISBN>1-861001-57-6</ISBN>
      <RecSubjCategories>
        <Category>Internet</Category>
        <Category>Web Publishing</Category>
        <Category>XML</Category>
    </RecSubjCategories>
    <Price>49.99</Price>
  </Book>
</Catalog>
```

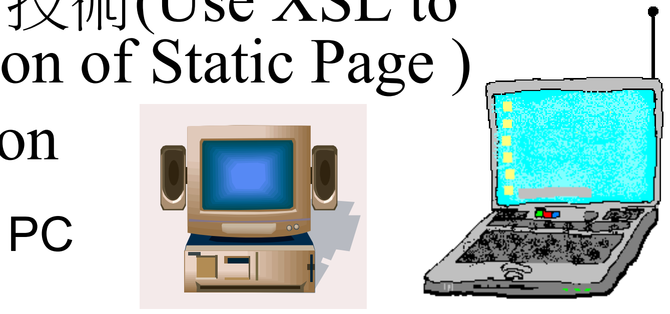
範例：DOM Tree

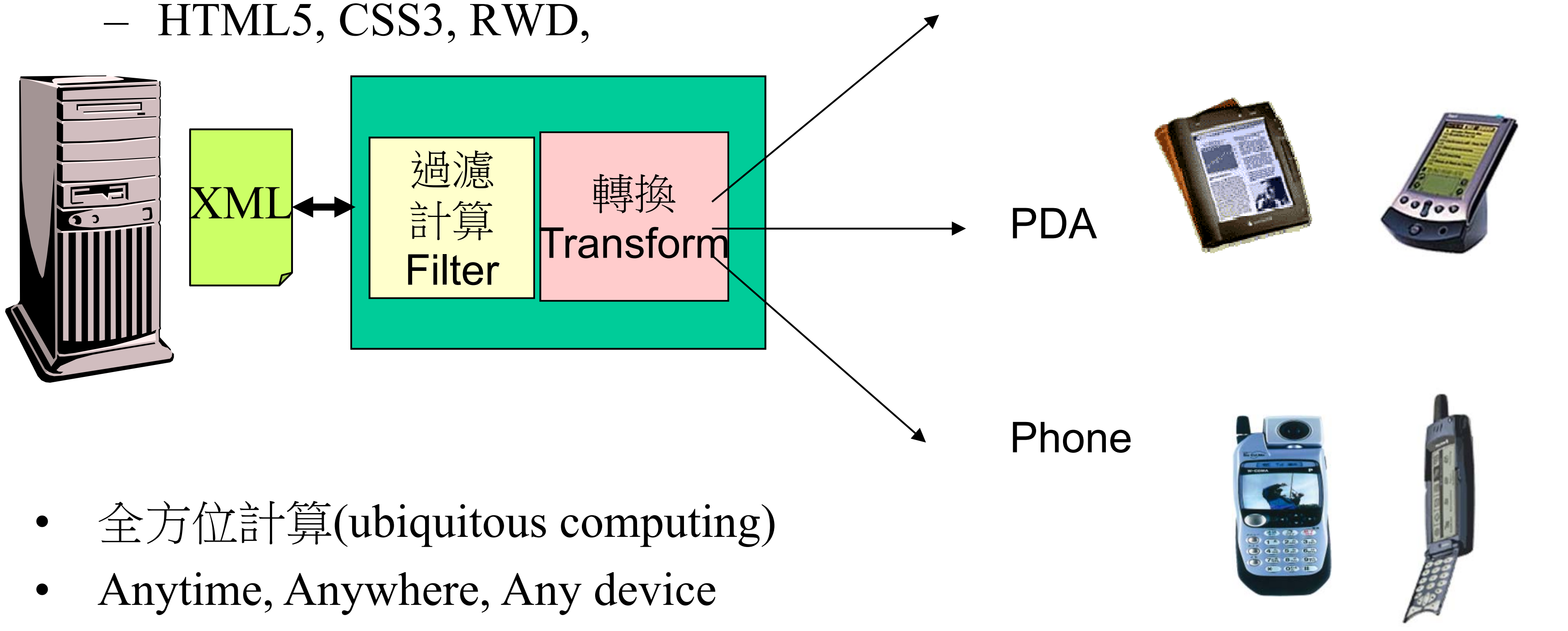


Document
Element :Catalog
Element : Book
Attribute : color
Element : Title
Element : Pages
Element : ISBN
Element : RecSubjCategories
Element : Price
Text : “red”
Text :“IE5 XML ..”
Text : ”480”
Text : ”1-8610..-6”
Text: “$49.95”
Element :Catagory
Element :Catagory
Element :Catagory
Text: “Internet”
Text: “Web Publishing”
Text: “XML”

DOM Design Pattern (1): Presentation Transformation

- 利用XSL來做呈現的轉換：靜態網頁技術(Use XSL to do presentation transformation; Extension of Static Page)
- Multi-modal Presentation Transformation
  - XSL, XHTML-Basic, SVG, VoiceXML
  - XML Viewer
  - HTML5, CSS3, RWD,

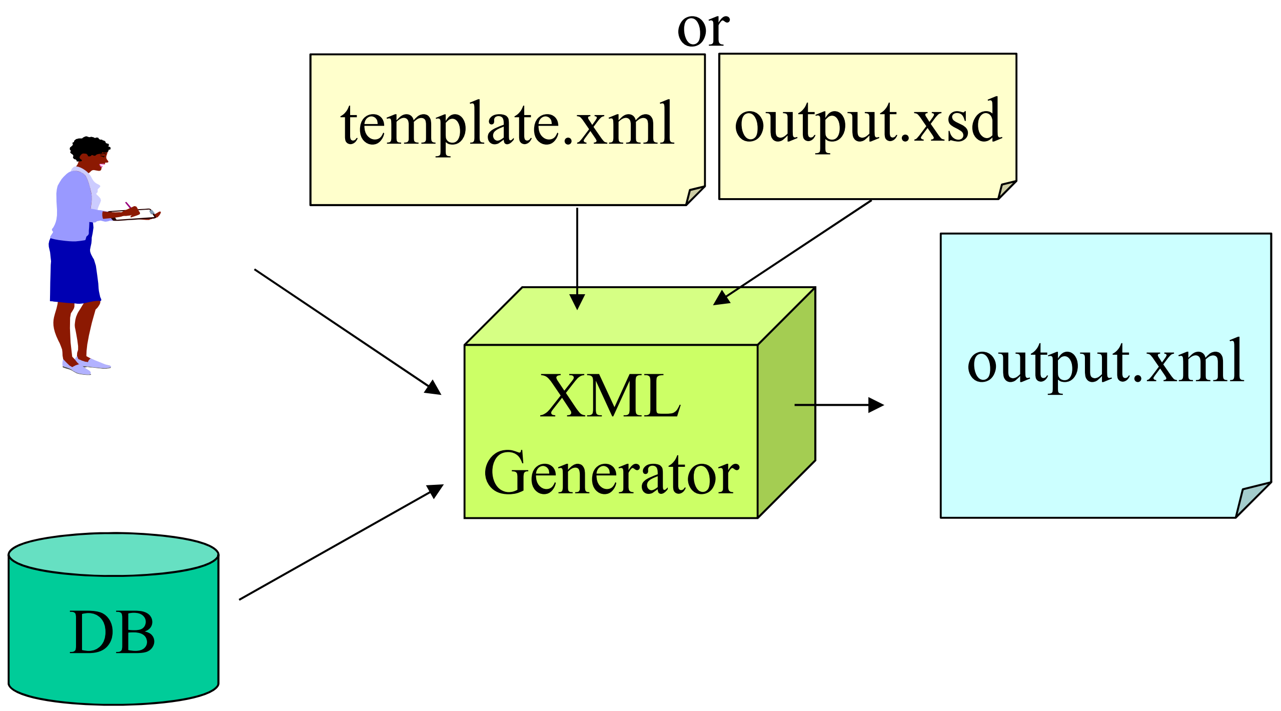




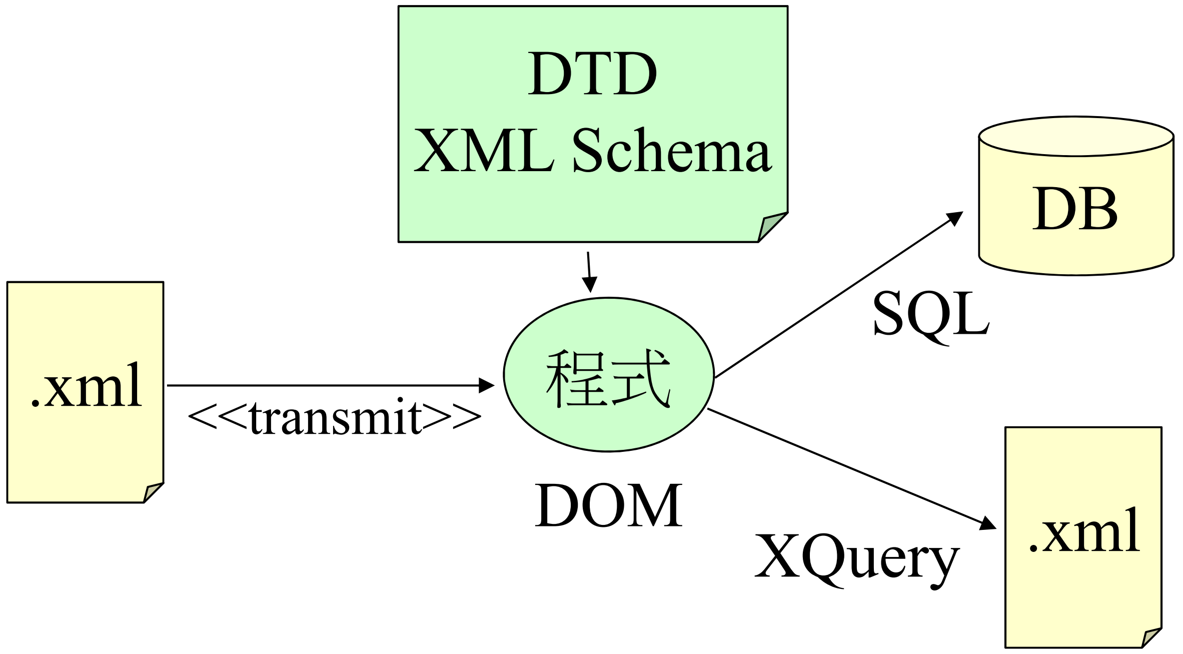
- 全方位計算(ubiquitous computing)
- Anytime, Anywhere, Any device

DOM Design Pattern (2): Template Transformation

- XML文件產生器(XML Generator)
  - 以Template檔來作為轉換的原始檔，利用使用者輸入資料或資料庫的資料做轉換作業(Use template file as the structure of the output file. Input data can be from user input or database

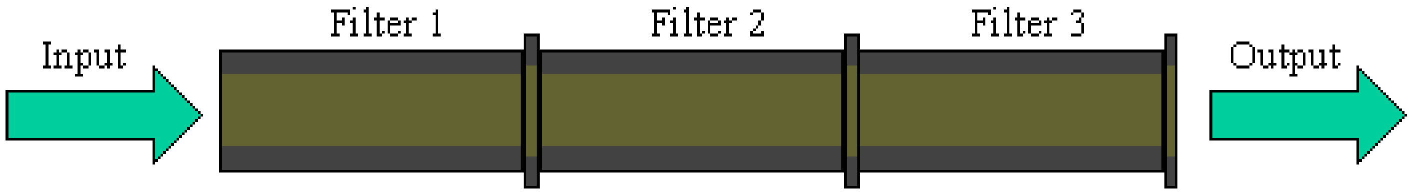


DOM Design Pattern (3): DB Converter



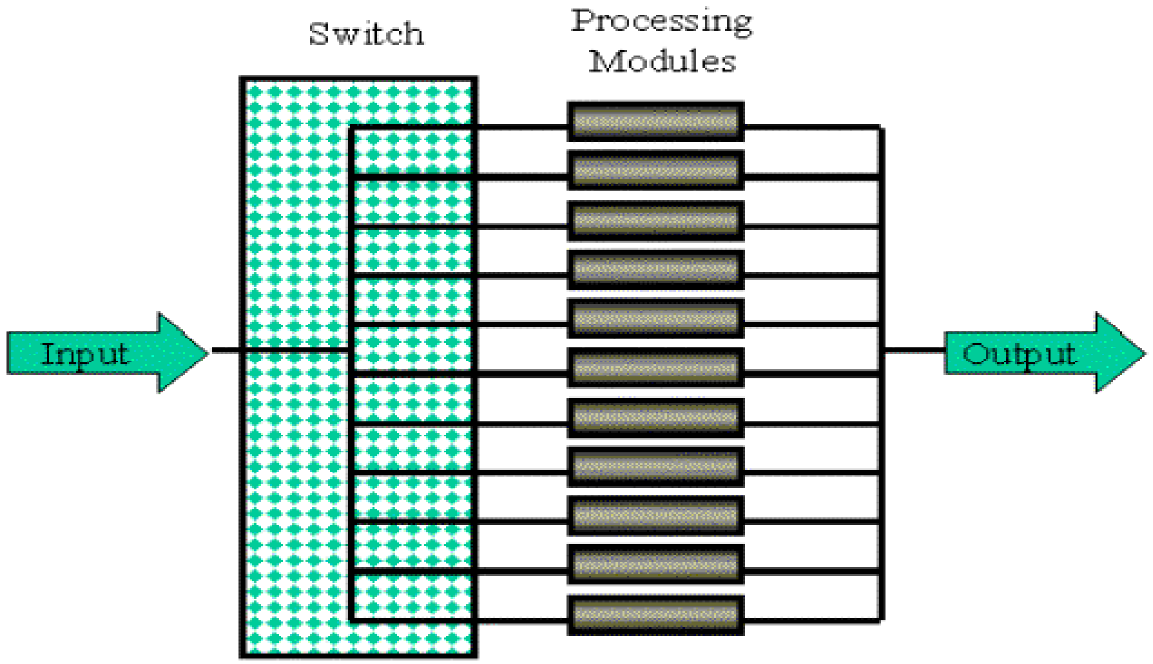
SAX Design Patterns:The Filter Pattern

- 又稱為 Called pipeline pattern.
  - –each stage of processing can be represented as a section of a pipeline.
- **Filter的工作：**
  - **Remove elements of the source document that are not wanted**
  - **Modify tags or attribute names**
  - **Perform validation**
  - **Normalize data values such as dates**



SAX Design Patterns:The Rule-Based Pattern

- 另一種SAX application的結構：Rule-based
  - 可以使modular simple and structured
- 使用 "Event-Condition-Action" model.



DOM API Object

- W3C DOM API
  - W3C官方定義(W3C Official Definition)
- Java DOM API(IBM, Oracle)
  - Java平台基礎(SUN，IBM)
  - JAXP(SUN)
  - Xerces(IBM):Aparche
- .NET DOM API (Microsoft)
  - ASP.NET
  - VB.NET
  - C#.NET
  - Visual Studio.NET

DOM API Object Hierarchy繼承圖

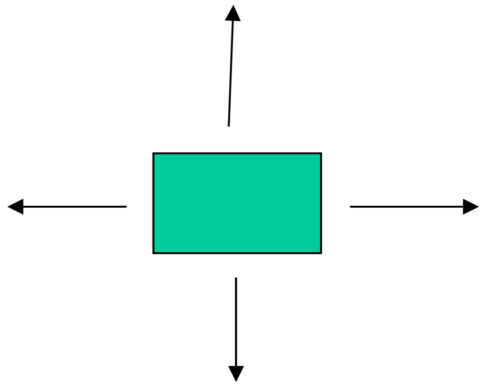
- **XMLDOMNode** (存取 XML data 的最基本介面 Basic interface to access XML data)
  - DOMDocument
  - XMLDOMDocumentFragment
  - XMLDOMAttribute
  - XMLDOMCharacterData
    - XMLDOMComment
    - XMLDOMText
      - XMLDOMCDATASection

繼承圖（二）

  - XMLDOMDocumentType
  - XMLDOMElement
  - XMLDOMEntity
  - XMLDOMEntityReference
  - XMLDOMNotation
  - XMLDOMProcessingInstruction
- **XMLDOMNodeList**
- **XMLDOMNamedNodeMap**
- **XMLDOMImplementation**

Accessing Nodes in DOM

- Walking the Tree.
  - parentNode().
  - firstChild().
  - nextSibling().
  - previousSibling().



- Accessing Nodes by Name:DFS
  - getElementsByTagName(***elementname***)

The Document Interface

- **Attribute**
  - doctype
  - implementation
  - documentElement

- **Method**
  - createElement()
  - createDocumentFragment()
  - createComment()
  - createCDATASection()
  - createProcessingInstruction()
  - createAttribute()
  - createEntityReference()
  - getElementsByTagName()

.NET DOM平台

- The Microsoft DOM Engine
  - ASP和C#為主要開發工具
  - client side: 適用於 IE5.0以上(內建)
  - server side: ASP2.0以上支援(IIS內建)
  - 在DocumentType interface 和中文處理上仍有一些問題
- 使用Microsoft XMLDOM ActiveX Data object.

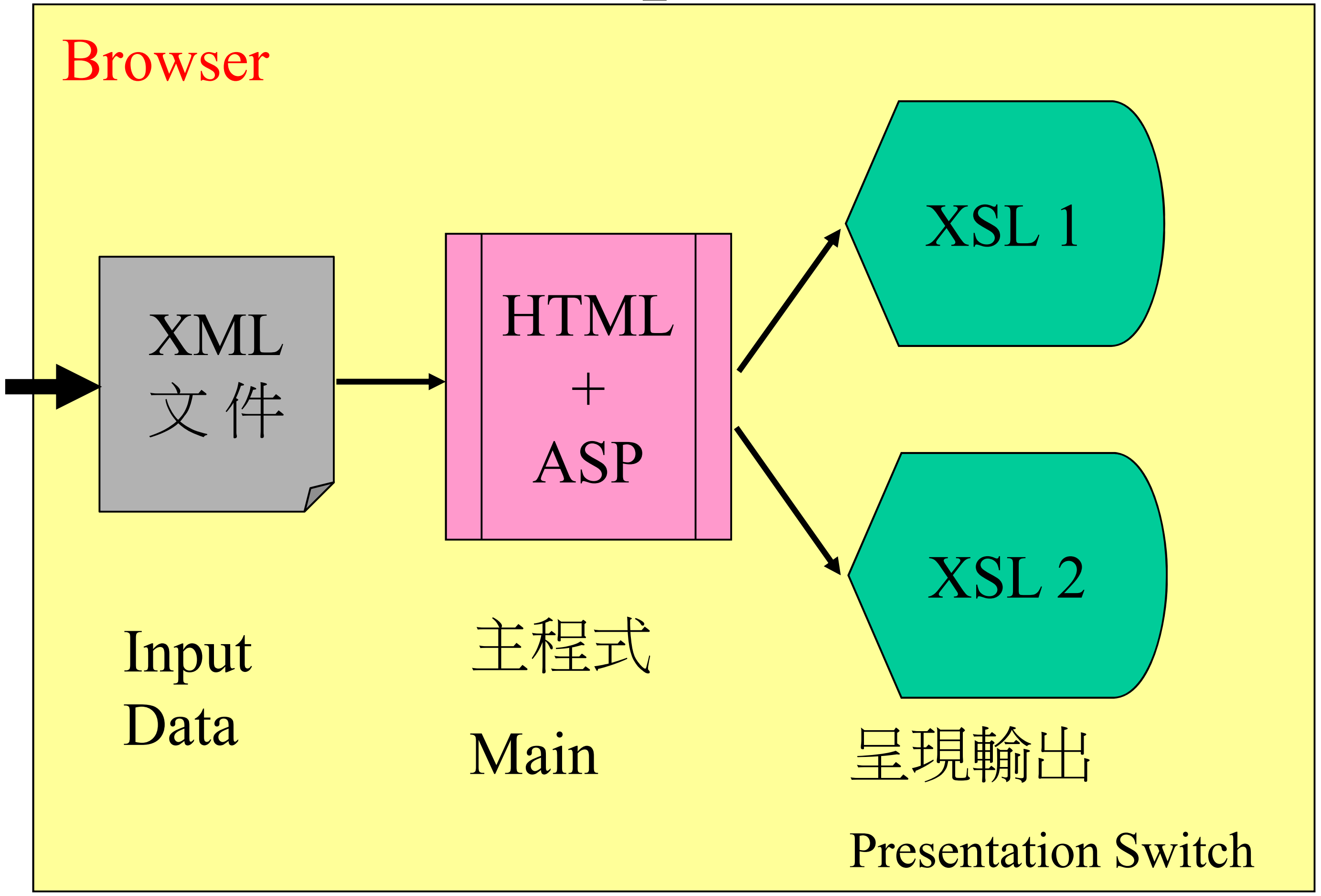
Java DOM平台

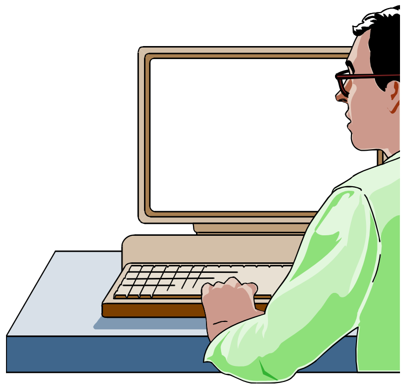
- Step1:安裝JDK
  - JDK1.4, JDK1.3.1
- Step2:安裝Java XML Parser(已內含在Aparche內)
  - JAXP (SUN)
  - Xerces (IBM)
- Step3:安裝Java Server Page Engine
  - Tomcat (aparche) (Web Server)
  - Resim (Web Server)
- 註：
  - Client side: Java application (JDK), Java applet (JVM)
  - Server side: JSP, Java servlet

DOM的Application開發工具

- Java
  - register the Java DOM engine as a class.
- C++ and C
  - (1) use the DLL provided by IBM.
  - (2) use the Microsoft ADO as COM.
- Visual Basic
  - (1) Set xmlobject = CreateObject(“Microsoft.XMLDOM”)
  - (2) xmlobject.load(file path)
- ASP
  - (1) Dim xmlobject
  - (2) set xmlobject = server.createObject(“MICROSOFT.XMLDOM”)

範例：配合ASP(XML Viewer)
Example: ASP DOM





範例：配合ASP：HTML 主程式

```
<html>
<head>
<title>產品搜尋結果</title>
</head>
<script>
var source;
var style;
var root;
var styleURL;
function init(){
	source = new ActiveXObject("Microsoft.XMLDOM");
	source.async = false;
	source.load("ex.xml");
	if (source.parseError.errorCode != 0){
		alert("Description: " + source.parseError.reason +
	"\nSource text: " + source.parseError.srcText);
	}
	root = source.documentElement;
	document.all.item("xslhead").innerHTML = "<h1>產品搜
	尋結果</h1><p><b>摘要：</b>" +
	root.selectSingleNode("//摘要").text + "</p>";
	style = new ActiveXObject("Microsoft.XMLDOM");
	style.async = false;
	styleURL = "ex1.xsl";
	changeXSL(styleURL);
}
```

HTML檔案(主程式)

```
function changeXSL(xsldoc){
		styleURL = xsldoc;
		style.load(styleURL);
		document.all.item("xslresult").innerHTML =
source.transformNode(style);
}function viewsrc(){
		alert(document.body.innerHTML);
}
</script>
<body onload="init();">
<div id="xslhead">
</div>
<hr>
<button onclick="changeXSL('ex1.xsl');">名稱排序
</button>
<button onclick="changeXSL('ex2.xsl');">價格排序
</button>
<button onclick="viewsrc();">檢視原始檔</button>
<hr>
<div id="xslresult">
</div>
</body>
</html>
```

範例：配合ASP:XML(input data)

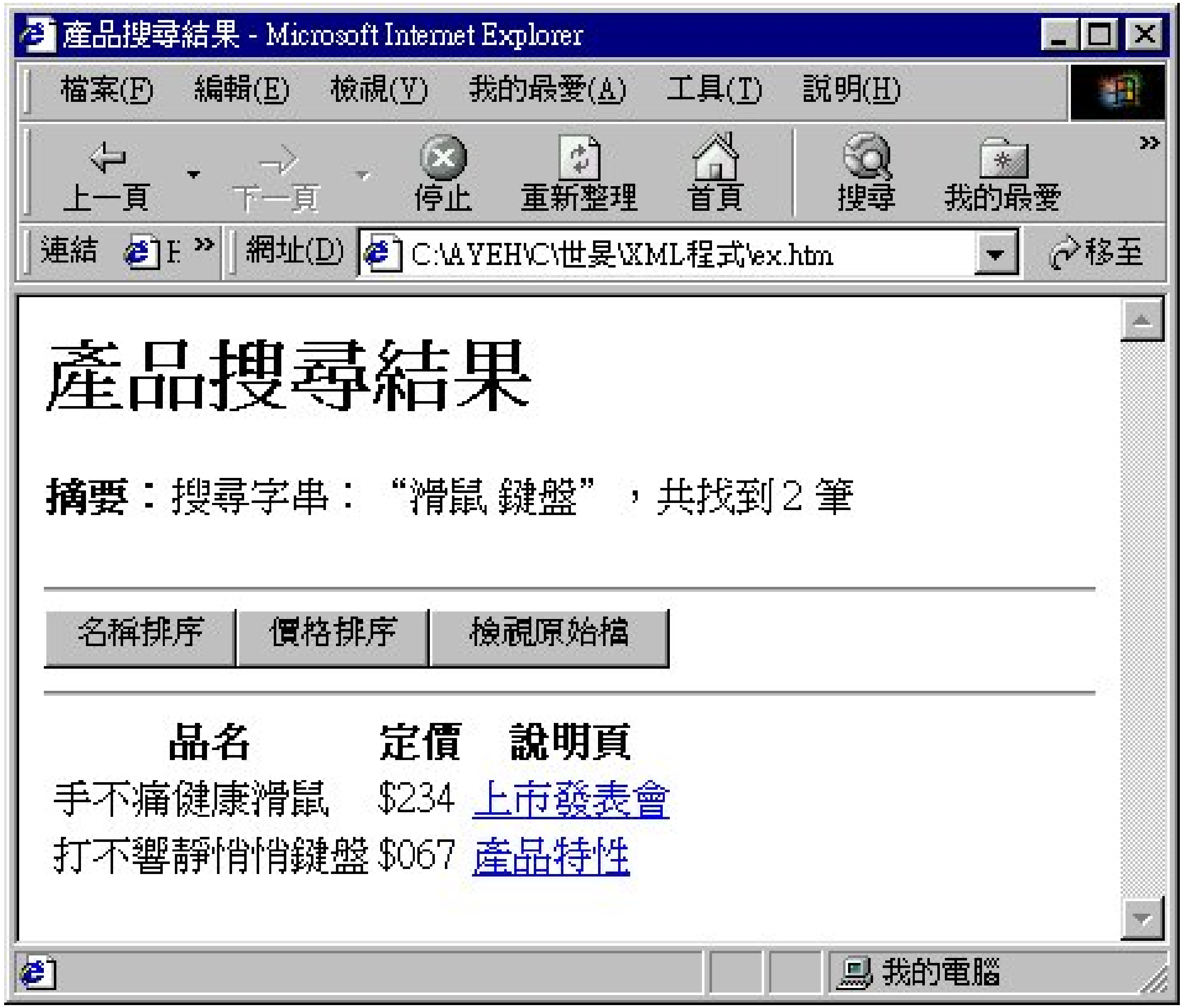
```
?xml version="1.0" encoding="Big5" ?>
<產品搜尋>
  <摘要>搜尋字串：“滑鼠 鍵盤”，共找到 2 筆</摘要>
  <產品>
    <貨號>12478943</貨號>
    <品名>手不痛健康滑鼠</品名>
    <定價>$234</定價>
    <說明頁 網址="http://foo.bar/mouse/12478943">上市發表會</說明頁
      >
  </產品>
  <產品>
    <貨號>83424723</貨號>
    <品名>打不響靜悄悄鍵盤</品名>
    <定價>$067</定價>
    <說明頁 網址="http://foo.bar/kbd/83424723">產品特性</說明頁>
  </產品>
</產品搜尋>
```

XML檔案

範例：配合ASP; XSL1

```
<?xml version="1.0" encoding="Big5" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
   <xsl:apply-templates select="產品搜尋"/>
</xsl:template>
<xsl:template match="產品搜尋">
  <table>
   <tr>
     <th>品名</th>
     <th>定價</th>
     <th>說明頁</th>
   </tr>
  <xsl:for-each select="產品" order-by="品名">
   <tr>
     <td><xsl:value-of select="品名"/></td>
     <td><xsl:value-of select="定價"/></td>
     <td><a><xsl:attribute name="href"><xsl:value-of select="說明頁/@網址
       "/></xsl:attribute><xsl:value-of select="說明頁"/></a></td>
   </tr>
  </xsl:for-each>
  </table>
</xsl:template>
</xsl:stylesheet>
```

範例：配合ASP; Display1


產品搜尋結果 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 下一頁 停止 重新整理 首頁 搜尋 我的最愛
連結 網址(D) C:\AYEH\C\世昊\XML程式\ex.htm 移至
我的電腦

產品搜尋結果

摘要：搜尋字串：“滑鼠 鍵盤”，共找到 2 筆

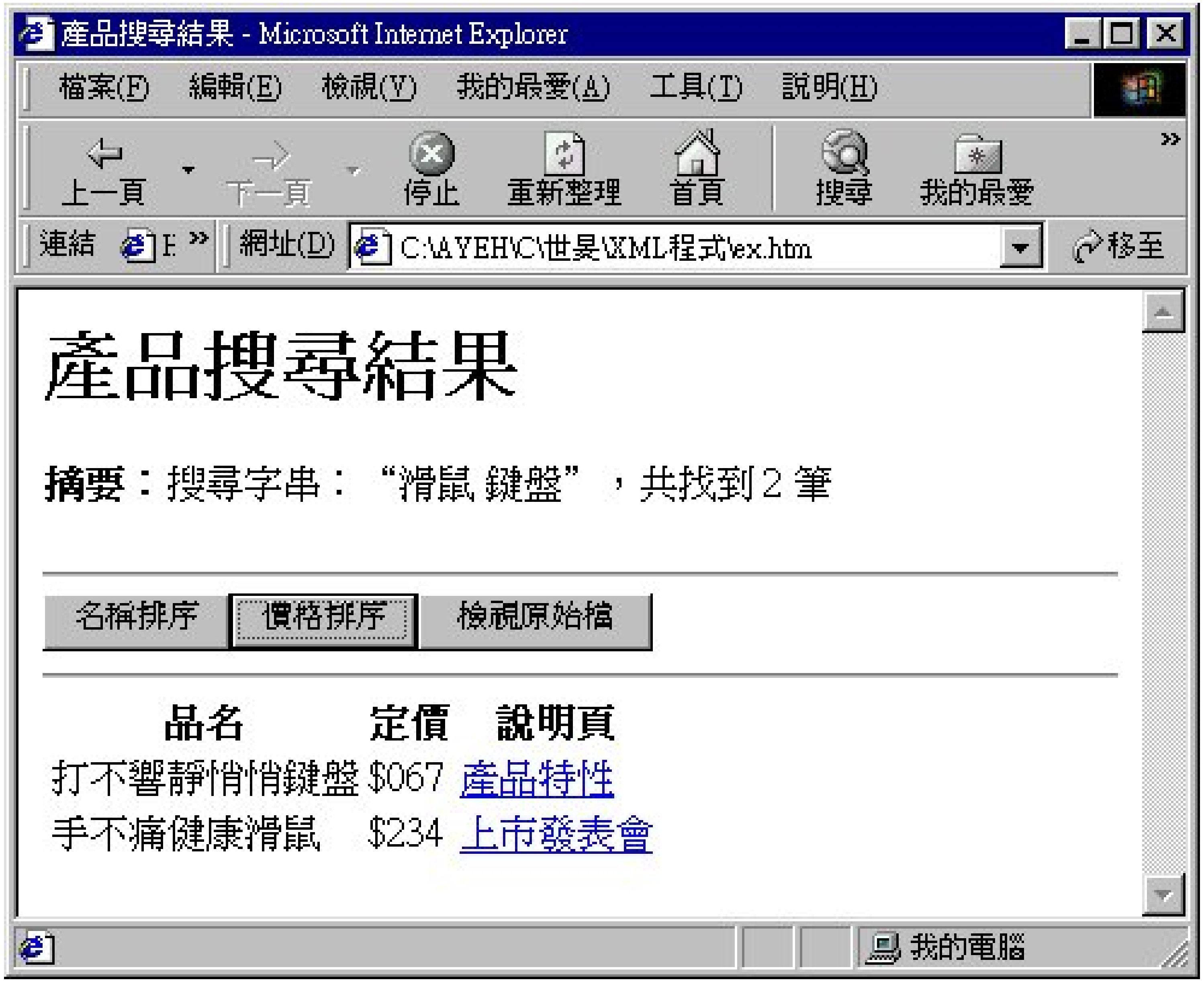
名稱排序 價格排序 檢視原始檔

品名	定價	說明頁
手不痛健康滑鼠	$234	上市發表會
打不響靜悄悄鍵盤	$067	產品特性

範例：配合ASP; XSL2

```
<?xml version="1.0" encoding="Big5" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
   <xsl:apply-templates select="產品搜尋"/>
</xsl:template>
<xsl:template match="產品搜尋">
  <table>
    <tr>
      <th>品名</th>
      <th>定價</th>
      <th>說明頁</th>
    </tr>
  <xsl:for-each select="產品" order-by="定價">
    <tr>
      <td><xsl:value-of select="品名"/></td>
      <td><xsl:value-of select="定價"/></td>
      <td><a><xsl:attribute name="href"><xsl:value-of select="說明頁/@網址
        "/></xsl:attribute><xsl:value-of select="說明頁"/></a></td>
    </tr>
  </xsl:for-each>
  </table>
</xsl:template>
</xsl:stylesheet>
```

範例： 配合ASP; Display2



產品搜尋結果 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 下一頁 停止 重新整理 首頁 搜尋 我的最愛
連結 網址(D) C:\AYEH\C\世昊\XML程式\ex.htm 移至
我的電腦

產品搜尋結果

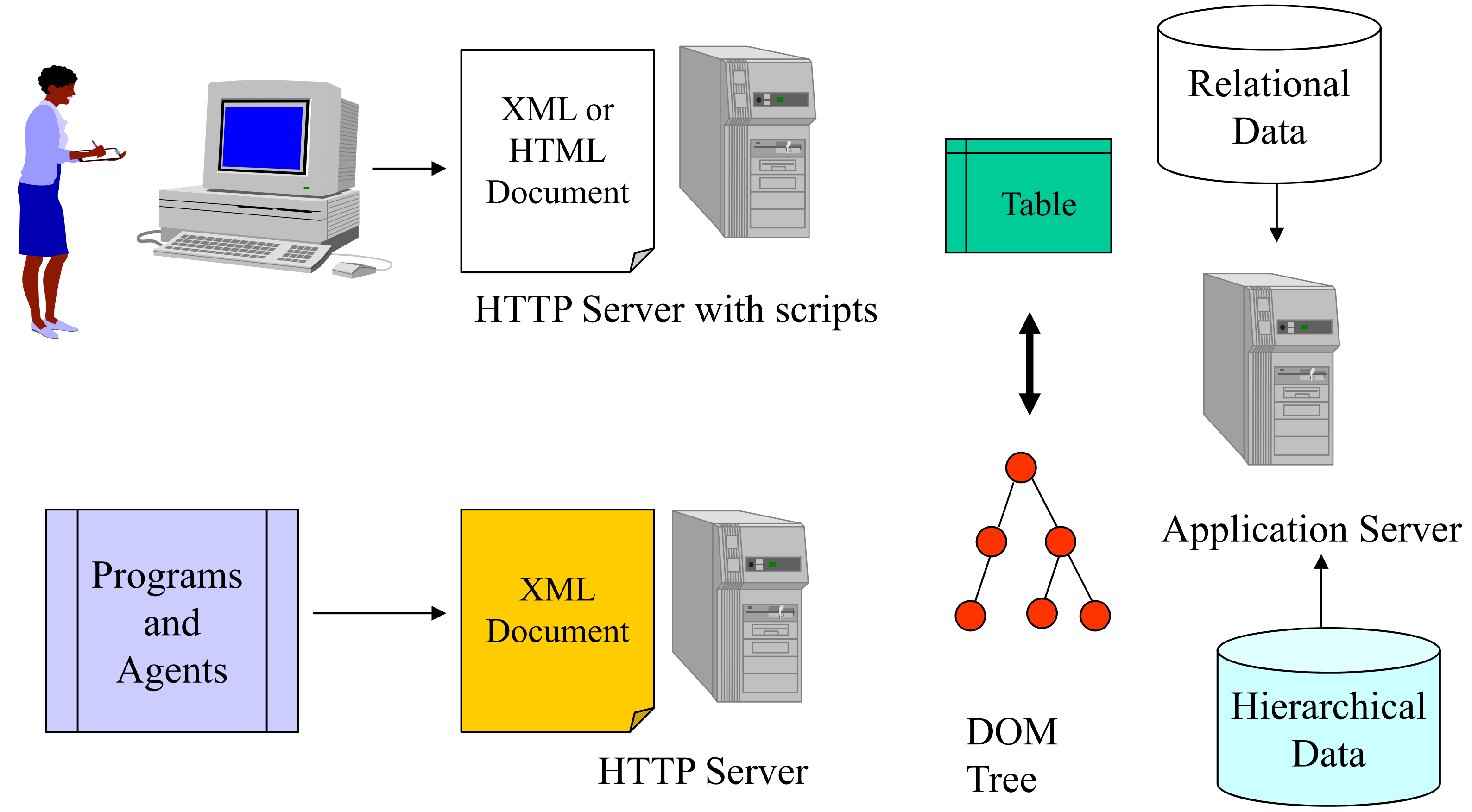
摘要：搜尋字串：“滑鼠 鍵盤”，共找到 2 筆

名稱排序 價格排序 檢視原始檔

品名	定價	說明頁
打不響靜悄悄鍵盤	$067	產品特性
手不痛健康滑鼠	$234	上市發表會

DOM和Database

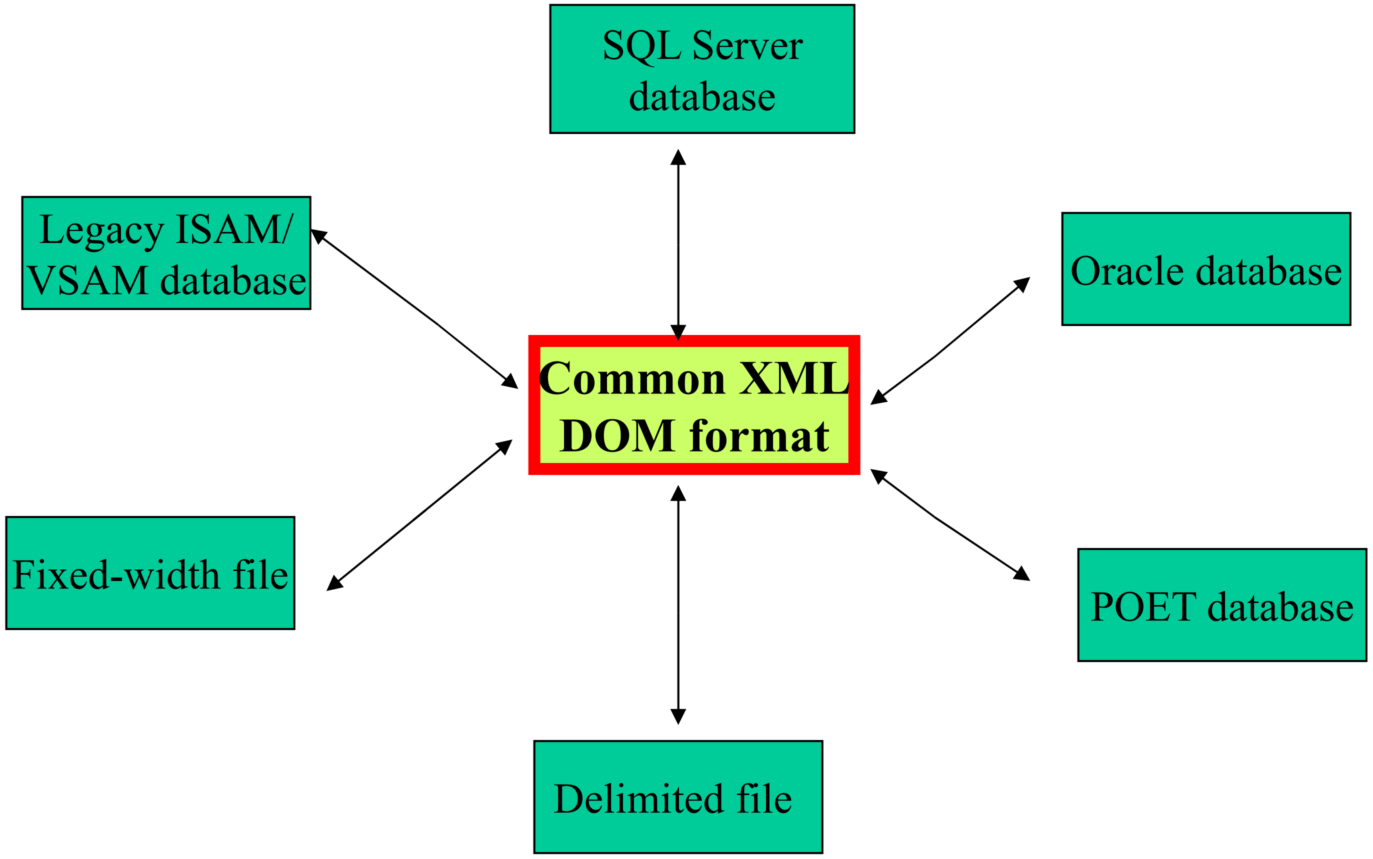
在伺服器端以DOM Tree和Database轉換為主要工作
(Server-side operations: DOM tree – Database transformation)



資訊系統做資料處理的三種儲存格式: XML, DB dump(B-Tree), Flat file

- 教務系統、會計系統等企業或組織營運系統需做年度備存作業，以迎接新年度營運。XML為三者中最理想儲存格式。
- Document Interchange (DB Interchange)
  - XML為platform-independent (DB dump NOT)
  - XML為self-describing (Flat data NOT)
  - XML為hierarchical information (Flat data NOT)
- Archiving
  - XML可以做為DB的archived information
  - 例如 Invoice DB的年度備份處理(annual archive)
- XML可長時間保存、跨平台、機器和人類可讀性

The DOM and Database



XML文件的管理問題：File System Shortages 的缺點

- Size：XML文件檔不能太大、搜尋慢(searching in XML large file is slow)
- Concurrency：多人同時修改文件會有問題(XML file can’t provide concurrent update)
- The right tool for the Job：XML Editor不足以提供文件各個部份的編輯 (tools for XML is incomplete)
- Versioning：很難做文件版本控制和追蹤(hard for versioning control)
- Security：多人對文件各部份有不同的權限，很難保護(security for different users in the same XML file)
- Integration: Centralization and Repetition

  不同文件有相同資料時的整合和控管

資訊系統類型 (Types of Information Systems)

- AP + DBMS+ DB(dump)

- AP + File System + XML

- AP + XML native DB(XML原生資料庫) + XML
- eXistDB, Path

Relational Database和XML Documents的差異 (Compare RDB with XML Documents)

- RDB rows have unique identifiers(primary key).
- RDB rows don’t imply sequence.
- RDB structures don’t provide hierarchical encapsulation.
- XML confounds attributes and text-only content.(後來有XML Schema定義datatype)
- XML allows a mixed content model for elements.(un-limited level tree structure)

DBMS如何使用XML (How DBMS use XML)

- 以XML文件做為DB的backup和archive (use as backup and Archive)
- 以XML文件做為不同DB之間資料轉移的package (Use as DB transit package)
- 以XML文件做為呈現DB資料的package (Use as the good web presentation of DB contents)
- 以XML文件做為一個DB本體(Use as the DB itself)(i.e. eXist DBMS)

XML和RDB的轉換一(mapping 1)：XML document文件←→One Table

- 一個DB只對應到XML文件的三層元素：(a DB can only map to three levels in a XML document)
  - DB Name
  - Table Name (1..n)
  - Column Name (1..m)
- Table Name (Relation)以下的子元素和參數全部展開成為Column Name (all the elements and attributes below level 3 should be extended in level 3)

XML對應一個Table
XML mapping to a Table

```
<?xml version="1.0"  encoding="BIG5"?>
<?xml-stylesheet type="text/xsl" href="9-9.xsl"?>
<Pdatabase>
<PersonData>
<name sex="男">楊胡炎</name>
 <birth>02/21/1947</birth>
 <occupation>高中老師</occupation>
 <spouse> <wife>賴美麗</wife></spouse>
 <interest>閱讀書籍、看電視、游泳、慢跑
       </interest>
<email>peter@hpdiy.zzn.com</email>
</PersonData>
<PersonData>
<name sex="男">吳煥崙</name>
 <birth>01/21/1946</birth>
 <occupation>國中老師</occupation>
 <spouse><wife>蔡麗華</wife> </spouse>
 <interest>閱讀書籍、睡覺</interest>
<email>allan@hpdiy.zzn.com</email>
</PersonData>
<PersonData>
<name sex="女">陳亞惠</name>
 <birth>03/11/1947</birth>
 <occupation>軍</occupation>
<spouse><husband> 黃承人</husband></spouse>
 <interest>看電視、游泳</interest>
 <email>yah@hpdiy.zzn.com</email>
</PersonData>
</Pdatabase>
```

- DB Name: Pdatabase
- Table Name: PersonData
- Column Name:
  - Name
  - Sex
  - Birth
  - Occupation
  - Spouse
    - Spouse type
    - Spouse name
  - Interest
  - email

XML對應一個Table
XML mapping to a Table

name	**sex**	**birth**	**occupation**	**Spouse Name**	**Spouse Type**	**interest**	**email**
楊胡炎	男	02/21/1947	高中老師	賴美麗	wife	閱讀書籍、看電視、游泳、慢跑	peter@hpdiy.zzn.com
吳煥崙	男	01/21/1946	國中老師	蔡麗華	wife	閱讀書籍、睡覺	allan@hpdiy.zzn.com
陳亞惠	女	03/11/1947	軍	黃承人	husband	看電視、游泳	yah@hpdiy.zzn.com

XML和RDB的轉換二(mapping 2)：XML文件←→Two Table

- RDB利用Join作業來結合不同Table的欄位 (RDB uses Join operation to join the columns in different table)
  - One-to-one Join
  - One-to-many Join
  - Many-to-many Join
- RDB必須加入Join的關鍵欄位Key (RDB should add the key column for join operation)

XML對應兩個Table
XML mapping to Two Tables

```
<OrderDB>
  <Order Orderid=" ">
    <BuyerID>
    <SellerID>
    <OrderTotal>
    <OrderDate>
    <Item>
      <ItemID>
      <ItemName>
      <OrderQuantity>
      <UnitPrice>
    </Item>
  </Order>
```

- DB Name: OrderDB
- 主檔(Master)：Order
  - OrderId
  - BuyerID
  - SellerID
  - OrderTotal
  - OrderDate
- 明細檔(Detail)：Item
  - OrderID
  - ItemID
  - ItemName
  - OrderQuantity
  - UnitPrice

XML對應兩個Table
XML mapping to Two Table

```
<OrderDB>
 <Order Orderid="1 ">
   <BuyerID>B001 </BuyerID>
   <SellerID>S001 </SellerID>
   <OrderTotal>12000 </OrderTotal>
   <OrderDate>2001/8/8 </OrderDate>
   <Item>
     <ItemID>X001 </ItemID>
     <ItemName>Pen </ItemName>
     <OrderQuantity>100 </OrderQuantity>
     <UnitPrice>100 </UnitPrice>
   </Item>
   <Item>
     <ItemID>X 004</ItemID>
     <ItemName>Scissor </ItemName>
     <OrderQuantity>10 </OrderQuantity>
     <UnitPrice> 100</UnitPrice>
   </Item>
   <Item>
     <ItemID>X010 </ItemID>
     <ItemName> Folder</ItemName>
     <OrderQuantity>5 </OrderQuantity>
     <UnitPrice>200 </UnitPrice>
   </Item>
 </Order>
 <Order Orderid="2 ">
   <BuyerID>B003 </BuyerID>
   <SellerID>S001 </SellerID>
   <OrderTotal>20000 </OrderTotal>
   <OrderDate>2001/8/9 </OrderDate>
   <Item>
     <ItemID>X002 </ItemID>
     <ItemName>Floopy Disk </ItemName>
     <OrderQuantity> 100</OrderQuantity>
     <UnitPrice>50 </UnitPrice>
   </Item>
   <Item>
     <ItemID> X004</ItemID>
     <ItemName>Scissor </ItemName>
     <OrderQuantity>50 </OrderQuantity>
     <UnitPrice>100 </UnitPrice>
   </Item>
   <Item>
     <ItemID>X010 </ItemID>
     <ItemName>Folder </ItemName>
     <OrderQuantity> 50</OrderQuantity>
     <UnitPrice>200 </UnitPrice>
   </Item>
 </Order>
</OrderDB>
```

XML對應兩個Table XML mapping to Two Table

- 主檔(Master)

OrderID	BuyerID	SellerID	OrderTotal	OrderDate
1	B001	S001	12000	2001/8/8
2	B003	S001	20000	2001/8/9

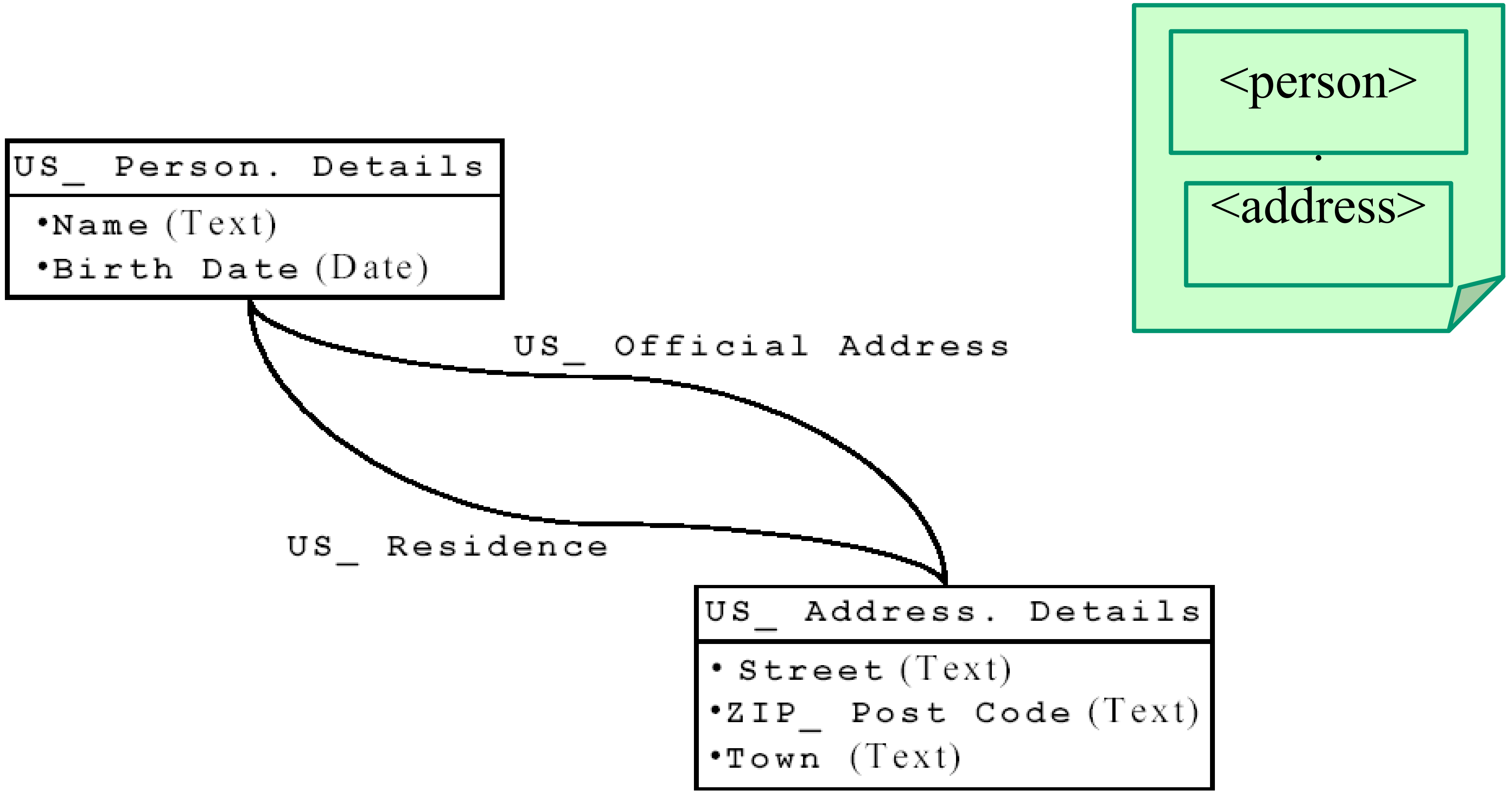
- 明細檔(Detail)

OrderID	ItemID	ItemName	OrderQuantity	UnitPrice
1	X001	Pen	100	100
1	X004	Scissor	10	100
1	X005	Folder	5	200
2	X002	Floopy Disk	100	50
2	X004	Scissor	50	100
2	X010	Folder	50	200

XML文件如何做最佳設計?
How do we design a good XML document

- RDB use 5 Normal Form Design; but how is XML document?
- Tag name與structure反映Semantics(Tag name and structure can reflect the semantics of the document)
- Use Object-Oriented Data Model Design
  - Class Diagram can reflect Semantics
- UN/CEFACT promote Core Component
- 處理XML文件最佳方式為OODBMS (The best way to store XML document is to use OODBMS)

XML 對 應 至 Object
(XML document mapping to Object



JSON

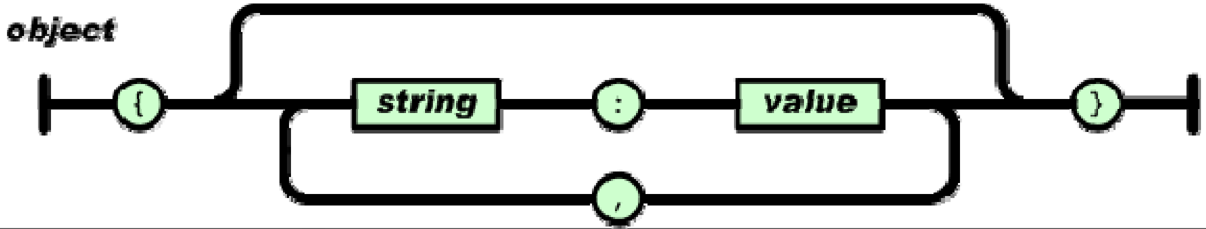
- **JSON**（**J**ava**S**cript **O**bject **N**otation）是一種由道格拉斯·克羅克福特Douglas Crockford 構想設計、輕量級的資料交換語言，以文字為基礎，且易於讓人閱讀。儘管JSON是Javascript的一個子集，但JSON是獨立於語言的文字格式，並且採用了類似於C語言家族的一些習慣。(JSON is a light-weight data exchange language)
- **NoSQL資料庫(NoSQL database)**
- 相對於傳統的關係型資料庫，一些基於文件儲存的NoSQL非關係型資料庫選擇JSON作為其資料儲存格式，比較出名的產品有：MongoDB、CouchDB、RavenDB等。

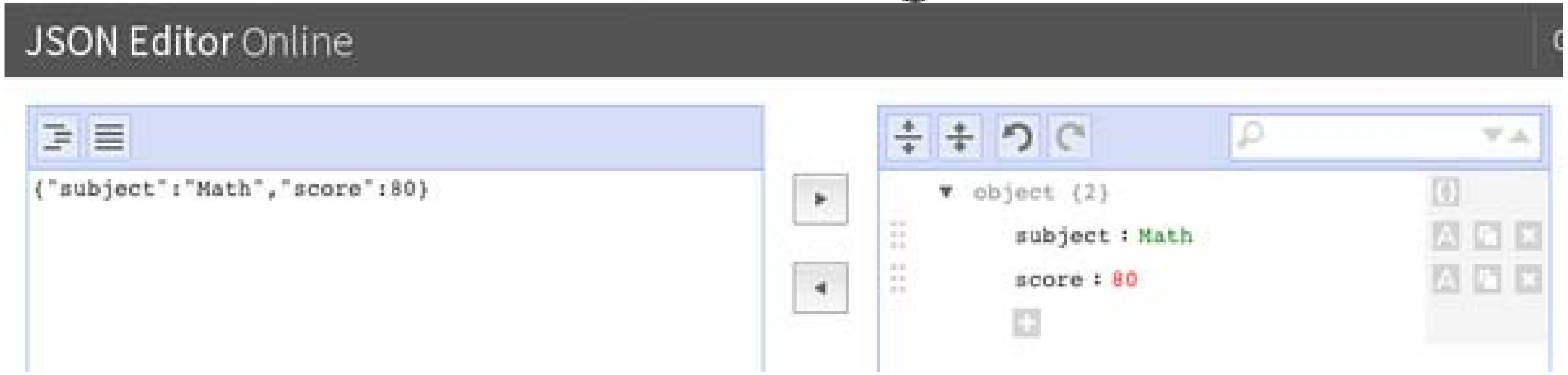
JSON格式

- 簡單來說，就是這二句重點：
- **物件(object)**用大括號 { }
- **陣列(array)**用中括號 [ ]
- 先記住這概念

物件(object)

- 是用key-value的方式儲存
- 範例(example)
- {"subject":"Math","score":80}
- key-value就是指一個鍵值(key)對應一個值(value)，跟變數很像
- 像是subject這個key有個值叫Math
- score它的值為80



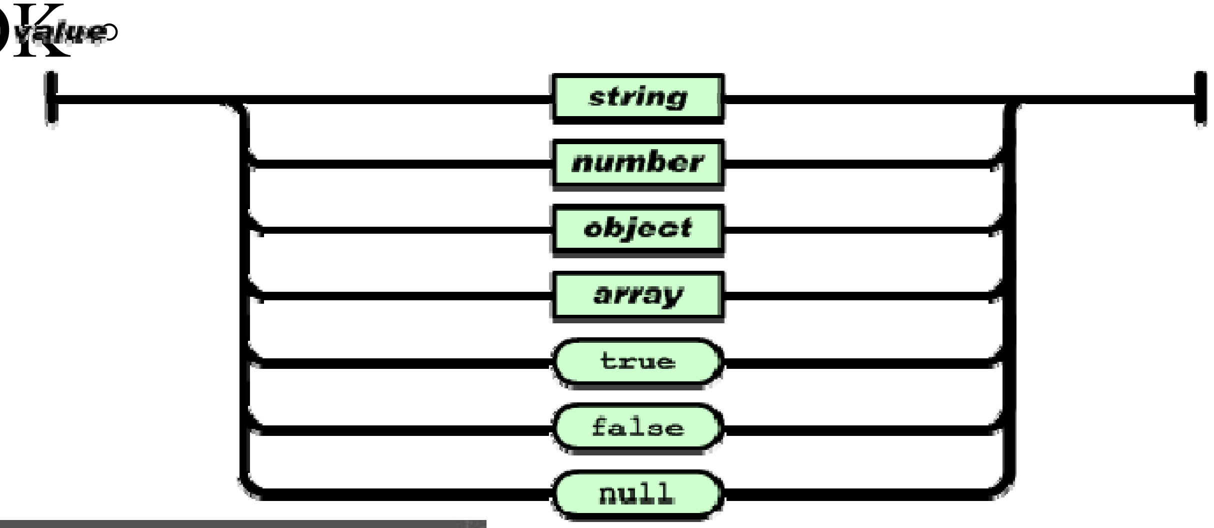


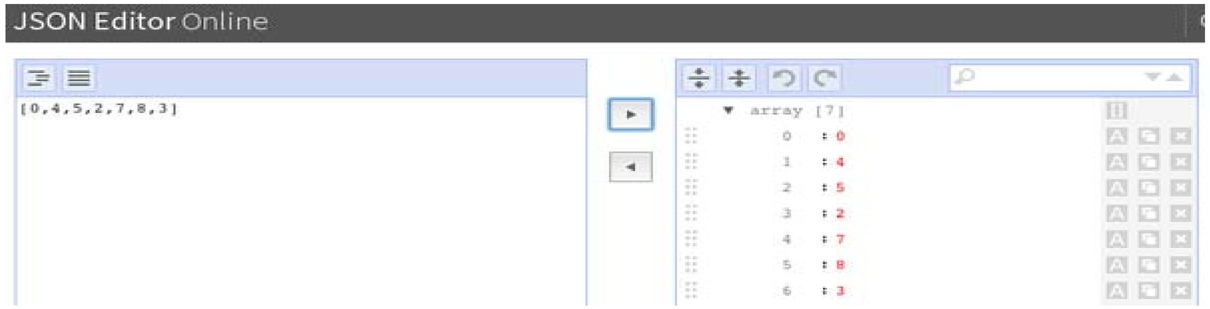
```
{"subject":"Math","score":80}
```

```
▼ object {2}
    subject : Math
    score : 80
```

陣列(array)

- 陣列可能就比較熟悉些, 例如[0,4,5,2,7,8,3]
- 這裡的範例是使用**數字**，但也可以是**文字**、**布林**或者是**陣列**、**物件**、**null**。當然，**混合**就不用說了，絕對OK。





```
[0,4,5,2,7,8,3]
```

```
▼ array [7]
    0 : 0
    1 : 4
    2 : 5
    3 : 2
    4 : 7
    5 : 8
    6 : 3
```

物件和陣列互相轉換 **Object and array exchange**

- 所以物件和陣列，某方面來說（不考慮資料損失），是可以**互相轉換**的(object and array can be exchanged)
- 若 物件object -> 陣列array
  - 就會損失鍵值(key)的資料，留下值(value)
    或是程式到時候指定說，要鍵值陣列(key array)，
    就會把所有的鍵值(key)合併一起成陣列
- 若 陣列array -> 物件object
  - 就可以將每個值編上數字
- 這裡注意一點
  json object的鍵值(key)，一定要用**文字**做鍵值

成績單(Transcripts)

- [{"name":"Tom","lastname":"Chen","report":[{"subject":"Math","score":80},{"subject":"English","score":90}]},{"name":"Amy","lastname":"Lin","report":[{"subject":"Math","score":86},{"subject":"English","score":88}]}]

- 成績單1

姓名 Name	Tom Chen
數學Math	80
英文Eng	90

成績單2

姓名 Name	Amy Lin
數學Math	86
英文Eng	88

XML的格式(XML format)

```
<data>
  <student>
    <name>Tom</name>
    <lastname>Chen</lastname>
    <report>
      <subject>
        <name>Math</name>
        <score>80</score>
      </subject>
      <subject>
        <name>English</name
>
        <score>90</score>
      </subject>
    </report>
  </student>
```

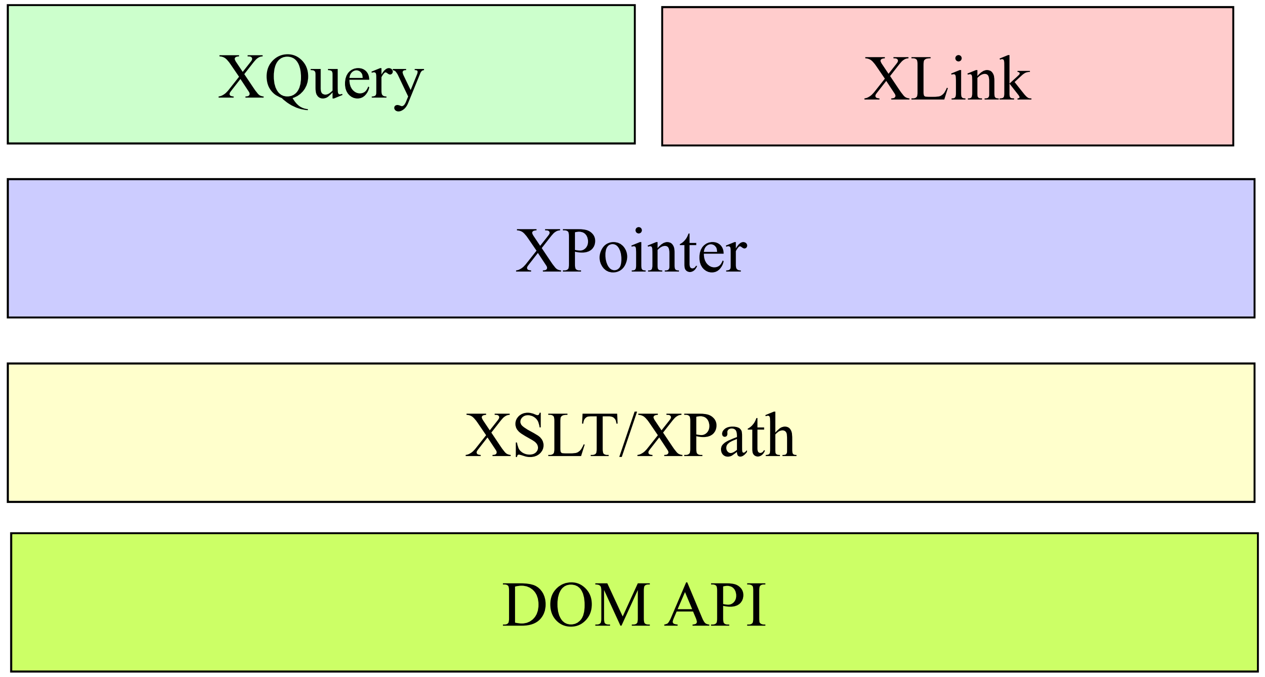
```
<student>
    <name>Amy</name>
    <lastname>Lin</lastname>
  <report>
     <subject>
        <name>Math</name>
        <score>86</score>
     </subject>
     <subject>
        <name>English</name>
        <score>88</score>
     </subject>
  </report>
 </student>
</data>
```

範例(JSON Format)

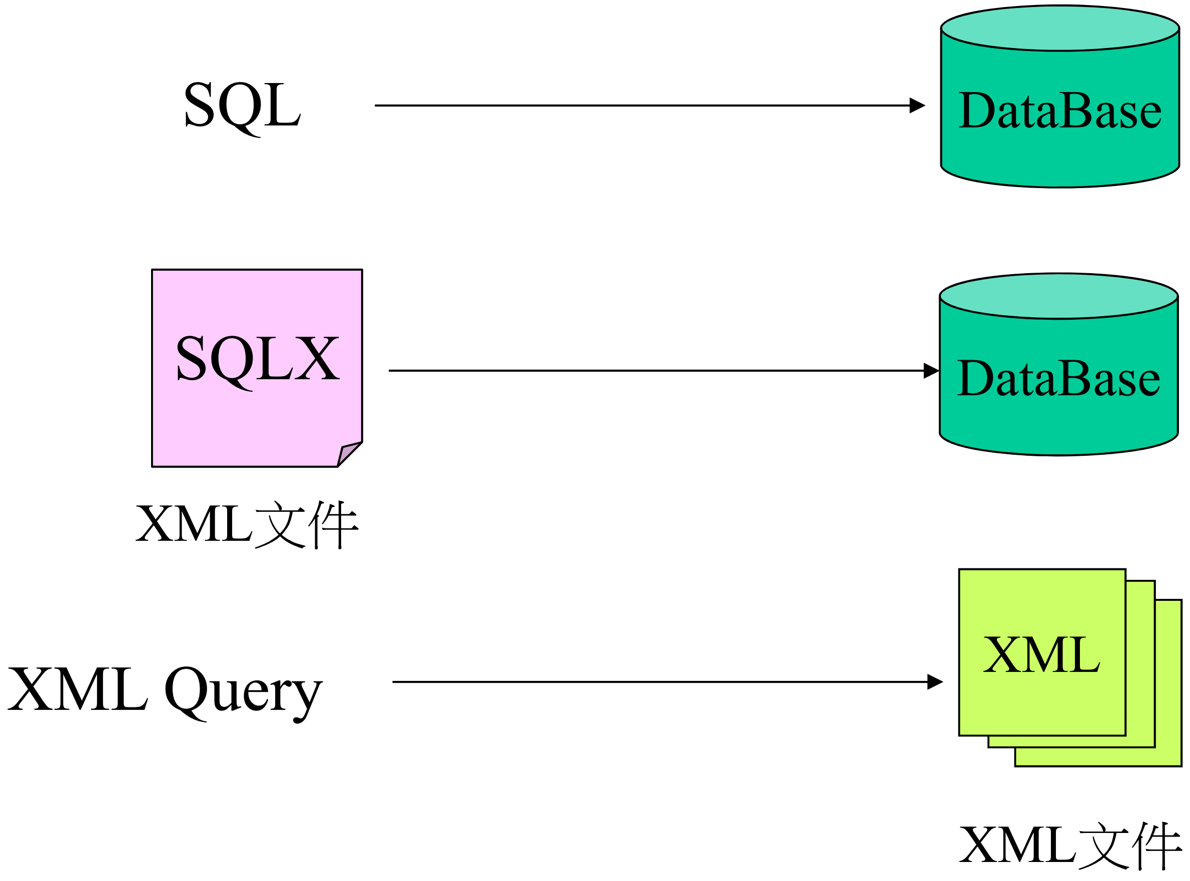
- {"name":"John Smith","age":32, "employed":true,“ address":{"street":"701 First Ave.","city":"Sunnyvale, CA 95125", "country":"United States"}, "children":[{"name":"Richard","age":7}, {"name":"Susan","age":4},{"name":"James","age" :3}]}

動態網頁未來方向：
XML Query and Processing

- XQuery有可能成為XML文件的”SQL語言”



Query Language



XML Query Language 的發展

- SQL
  - Relational Database Standard Query Language
  - 1974 SEQUEL by Chamberlin and Boyce; ISO
- XML-QL: by IBM(Notes;1998/8/19)
  - 查詢方式類似SQL的語法
- XSLT and XPath
  - 結構化文件(樹狀)的查詢方式
  - Recommendation
- XQL：by Microsoft(Notes;1998/11/6)
  - 結構化文件(樹狀)的查詢方式(XSLT-XPath的擴充)
- XSQL：by Oracle
- XML-Query Data Model
  - Working Draft : 2000/5/11,2000/8/15
- XQuery
  - Working Draft: 2001/6/8, 2003/11/12
  - Candidate Recommendation 2005/11/3
  - Recommendation 2007/1/23

SQL範例

```
SELECT Book.Title, RecSubjCategories.Category
  FROM Book INNER JOIN RecSubjCategories
    ON Book.BookID=RecSubjCategories.BookID
  WHERE Book.Author="Kevin Williams"
```

Book
RecSubjCategories

XML-QL範例

```
CONSTRUCT <Titles> {
  WHERE
    <Book>
      <Title>$t</Title>
    </Book> IN “http://www.wrox.com/XML/catalog.xml”
  CONSTRUCT
      <Title>$t</Title>
}</Titles>
```

Result Document ➡

```
<Titles>
  <Title>IE5 XML Programmer’s Reference</Title>
  <Title>Designing Distributed Applications</Title>
</Titles>
```

XQL

- Wildcards(*)

Example : ‘movies/*/title’

Result :

```
<xql:result>
        <title> Raising Arizone </title>
        <title> Midnight Run </title>
        <title> The Usual Suspects </title>
         <title> The Abyss </title>
      < /xql: result>
```

XQuery的文件(2001/6/8)

- **XML Query Use Cases**
- **XQuery 1.0 and XPath 2.0 Data Model**
- **XQuery 1.0 Formal Semantics**
- **XQuery 1.0: An XML Query Language**
- **XML Syntax for XQuery 1.0 (XQueryX)**

XQuery

- **XQuery 1.0: An XML Query Language:Recommendation 23 January 2007**
- **XML Syntax for XQuery 1.0 (XQueryX):**

W3C Recommendation 23 January 2007

- **XML Query Use Cases: W3C Working Group Note 23 March 2007**

XQuery語法

- XQuery選取與過濾元素一共有兩種方式，
- 一種為路徑運算式，不過此方式過於簡單，因此只能選取元素或屬性而已；
- 另一種則是使用FLWOR運算式，FLWOR是一種功能更為強大的運算式。FLWOR運算式主要是由for、let、where、order by和return子句所組成

FLWOR運算式

- For子句：可以將in指令後的路徑運算式依序變數取得的順序，指定給in前的變數，每次一個項目，直到順序的最後一個項目為止。
- Let子句：用來指定XQuery變數的值，變數值可以是項目或順序。
- Where子句:指定條件運算式來進一步過濾查詢結果，只有當運算式為true時，才執行return子句。
- Order by子句：可以指定輸出結果的排序方式。除此之外，我們可以使用「,」符號指定多個排序方式。
- Return子句：輸出查詢的結果，如果是使用路徑運算式，則就是輸出選取的節點內容。

XQuery範例: XML文件

```
<bib>
 <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price> 65.95</price>
 </book>
 <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
     <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
     <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price> 39.95</price>
  </book>
  <book year="1999">
    <title>The Economics of Technology and Content for Digital TV</title>
    <editor> <last>Gerbarg</last><first>Darcy</first> <affiliation>CITI</affiliation>
    </editor>
    <publisher>Kluwer Academic Publishers</publisher>
    <price>129.95</price>
  </book>
</bib>
```

XQuery範例: Query 1

- List books published by Addison-Wesley after 1991, including their year and title.
- ***Solution in XQuery:***

```
<bib>
  {
   FOR $b IN document("http://www.bn.com")/bib/book
   WHERE $b/publisher = "Addison-Wesley"
               AND $b/@year > 1991
   RETURN
     <book year={ $b/@year }> { $b/title } </book>
  }
</bib>
```

XQuery範例: Query1結果

Expected Result:

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix
           environment</title>
  </book>
</bib>
```

XQuery範例: Query 2

- For each book in the bibliography, list the title and authors, grouped inside a "result" element.
- ***Solution in XQuery:***

```
<results>
 { FOR $b IN document("http://www.bn.com")/bib/book
   RETURN <result> { $b/title }
   {  FOR $a IN $b/author
      RETURN $a }
      </result> }
</results>
```

XQuery範例: Query2結果

Expected Result:

```
<results>
  <result>
    <title>TCP/IP Illustrated</title>
    <author> <last>Stevens</last> <first>W.</first> </author>
  </result>
  <result>
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last> <first>W.</first> </author> </result>
    <result><title>Data on the Web</title>
    <author><last>Abiteboul</last> <first>Serge</first> </author>
    <author> <last>Buneman</last> <first>Peter</first> </author>
    <author> <last>Suciu</last> <first>Dan</first> </author>
  </result>
  <result>
    <title>The Economics of Technology and Content for Digital TV</title>
  </result>
</results>
```

- <bib> { for $b in doc("http://bstore1.example.com/bib.xml")/bib/book where $b/publisher = "Addison-Wesley" and $b/@year > 1991 return <book year="{ $b/@year }"> { $b/title } </book> } </bib>

XML 原生資料庫

- eXist Open Source DBMS use XQuery
- http://exist.sourceforge.net/

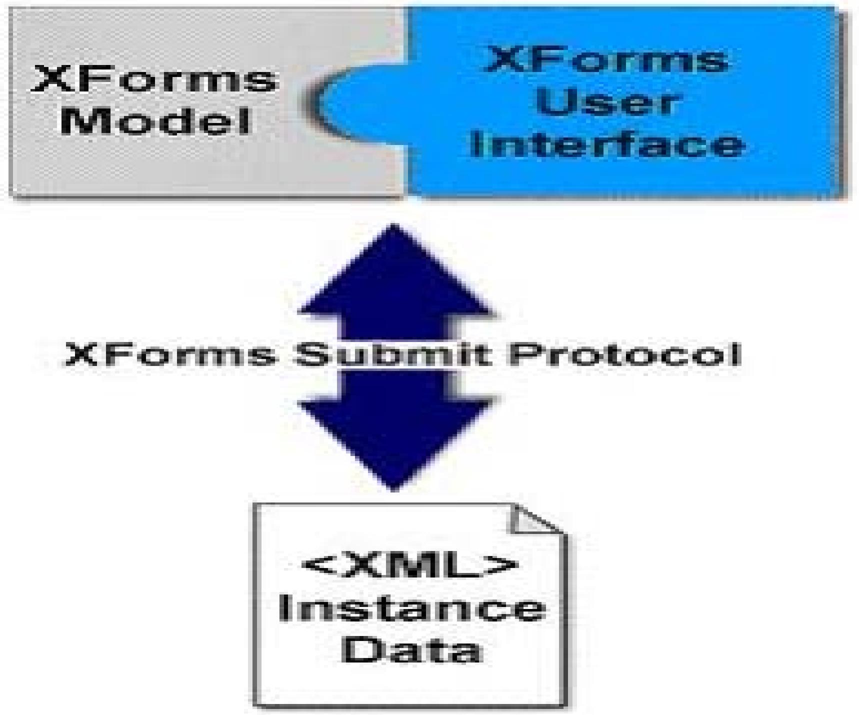
XForms

- **XForms** is an XML format for the specification of a data processing model for XML data and user interface(s) for the XML data, such as web forms.

XForms

- XForms Recommendation 2003/10/14
- XForms 1.0 (Third Edition) : W3C Recommendation 29 October 2007
- XForms 1.1 : W3C Recommendation 20 October 2009
- Implementations
  - X-Smiles: http://www.x-smiles.org/
  - 中文處理不完備
  - MS的IE Browser不支援

XForms Framework

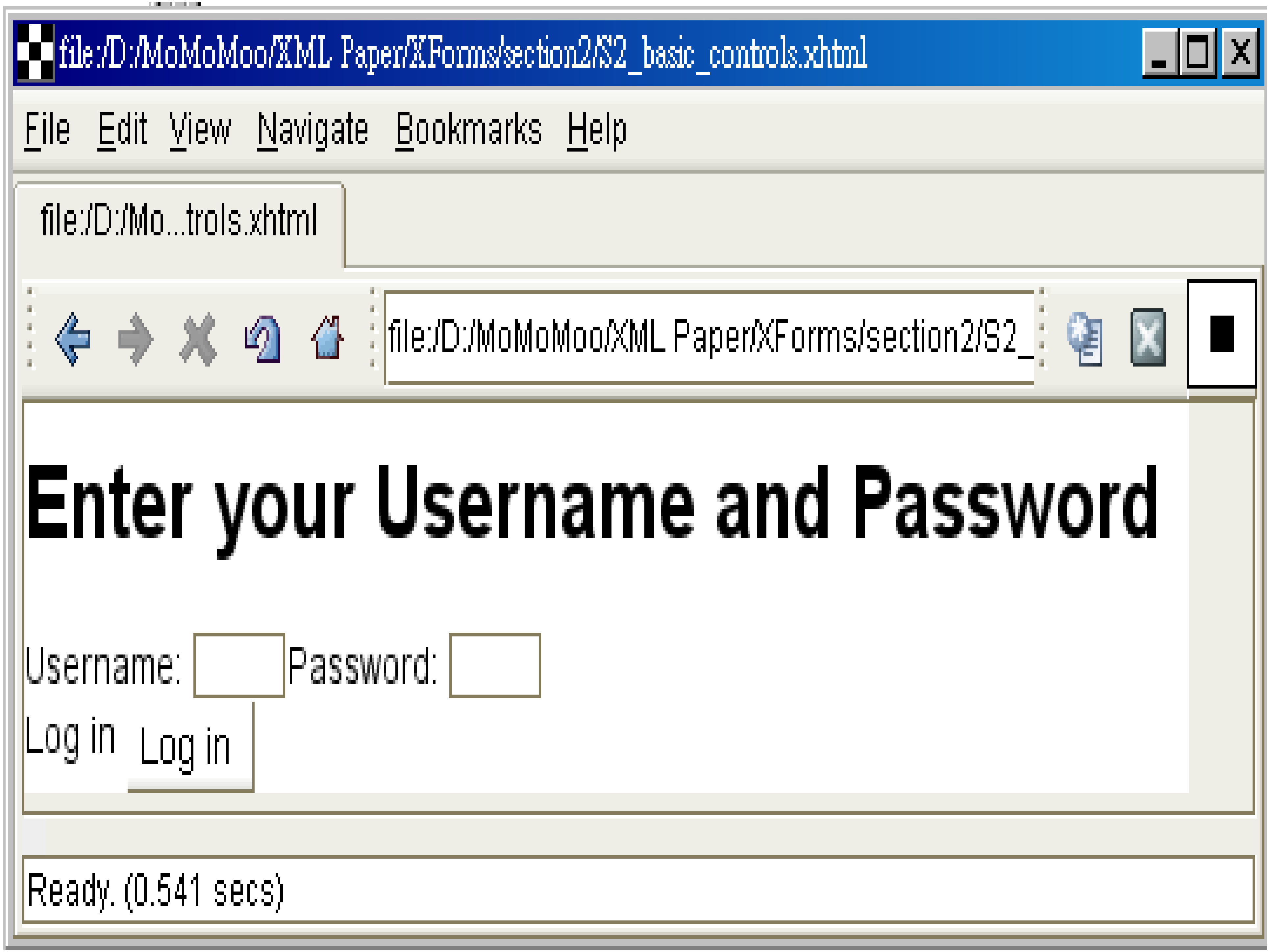


The instance document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xforms="http://www.w3.org/2002/xforms/cr">
<head>
      <xforms:model>
            <xforms:submission action="http://www.example.com" method="post" />
                        <xforms:instance xmlns="">
                                    <logininfo>
                                                <username />
                                                <password />
                                    </logininfo>
                        </xforms:instance>
      </xforms:model>
</head>
      <body>
            <h1>Enter your Username and Password</h1>
            <p />
      </body>
</html>
```

Basic control

```
<body>
    <h1>Enter your Username and Password</h1>
    <p>
            <xforms:input ref="/logininfo/username">
                        <xforms:label>Username: </xforms:label>
            </xforms:input>
            <xforms:input ref="/logininfo/password">
                        <xforms:label>Password: </xforms:label>
            </xforms:input>
            </p>
            <xforms:submit>
                        <xforms:label>Log in</xforms:label>
            </xforms:submit>
    </p>
</body>
```


file:/D:/MoMoMoo/XML Paper/XForms/section2/S2_basic_controls.xhtml

File
Edit
View
Navigate
Bookmarks
Help
file:/D:/Mo...trols.xhtml
file:/D:/MoMoMoo/XML Paper/XForms/section2/S2_

Enter your Username and Password

Username: Password:

Log in

Ready. (0.541 secs)

Submitting the form

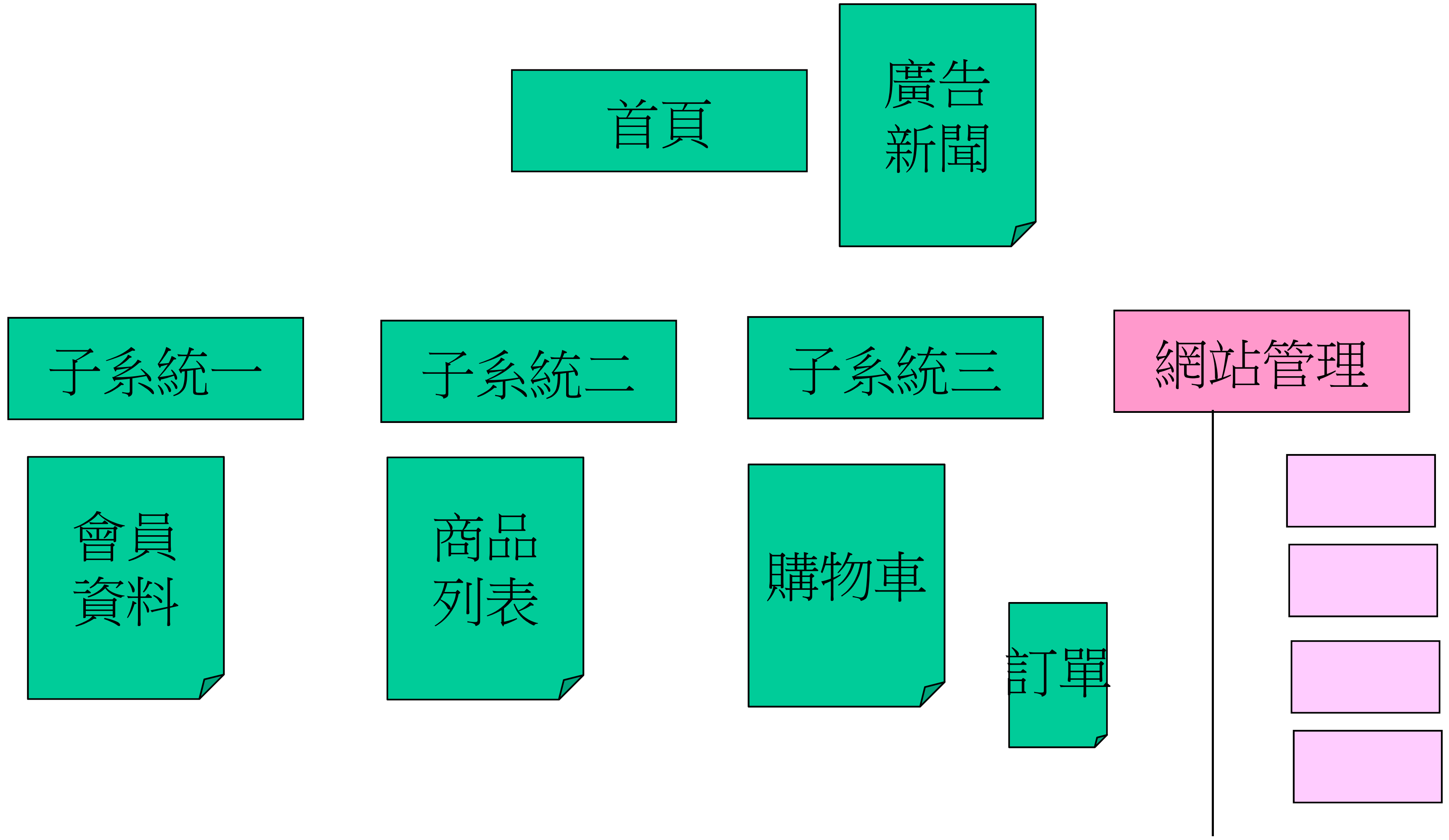
- XForms Output: sample login form

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<logininfo>
    <username>nick</username>
    <password>supersecret</password>
</logininfo>
```

XML動態網頁實習

- XML Viewer
  - Data Island: XML與XSL動態結合
- XML Generator
  - Flat data transform to XML file
  - XForms generate XML instance
  - XQuery generate XML results
- DB Transformer
  - XML to MySQL (or Access) (1table, or 2table)
  - MySQL (or Access) to XML

XML動態網頁做網站管理



總結

- 動態XML網頁技術：
  - DOM, XQuery, XForms
  - DTD, XML Schema
  - Java, .NET
- DOM將為所有電腦網路應用的統一內部資料結構
- DOM API目前為Programmer要發展電腦網路應用必備的核心技術
- DOM Tree的各節點在Level 3以後會以data type編碼

複習(Review)

- 動態網頁情境(XML Dynamic Page Mechanism)
- 動態網頁(XML Dynamic Page Design Patterns)
- 比較Tree-based Parser 和 Event-driven Parser(Compare Tree-based Parser with Event-driven Parser)
- XML和DB Table轉換類型 (2 types of transformation scheme from XML to DB tables)
- XML 和RDB比較 (Compare XML and Relational DB)
- 何謂DOM (What is DOM)
- 何謂XQuery (What is XQuery)
- 何謂XForms (What is XForms)

Homework

- Take a e-Commerce Website (Amazon, eBay, 淘寶網, PCHome)as an example, generate as least two order forms,

(a). Produce your own order.xml

(b). Based on the order.xml, generate the DTD

(c). Generate the XML schema, XSD

(d). Illustrate how to arrange DB tables to store the order.xml data and its related information