植基於 XML 技術之電子商務表單驗證框架

游峰碩
崑山科技大學資訊管理系 助理教授
e-mail：fsyu@mail.ksu.edu.tw

摘要

傳統上，電子商務網站之表單資料驗證邏輯大多是與程式碼寫在一起，一但企業處理的邏輯更改了，程式碼也必須跟著修改。這一點增加了程式的複雜度並且也降低了系統的維護性(Maintainability)。本論文採用物件導向分析與設計(OOAD)之方法，提出一個植基於 XML 技術之表單驗證框架(form validation framework)。此框架之優點在於使用者可以在不修改程式碼的情況下，根據檢驗需求，更改檢驗條件，此點大幅提升了程式的可維護性，並且藉由框架設計的優點，也提供了該軟體元件在未來的擴充性(Extensibility)。另外，對於採用早期 CGI 技術建置之網站，在有限的成本、時間、金錢與其他資源的考量下；或者是不想全面重新改寫系統，但又希望能夠運用新的技術所帶來的優點的話，本研究提供了一個有效的重構解決方案。

關鍵詞：XML，物件導向分析與設計，框架，表單，重構。

Abstract

Traditionally, the logic for form validation process is coded together with program code. Therefore, once the business logic changed, the program code need to be modified accordingly. This not only increases the complexity of the program, but also degrades the maintainability of the system. By using the methodology of Object-Oriented Analysis and Design, and XML technology, this paper proposes a software framework to simplify form validation task. By using this framework, users no longer need to modify the program when the business logic changes. This can greatly enhance the maintainability of the program, and provide the extendibility of the system in the future. On the other hand, under the consideration of cost, time, money and resource constraint or wish to take advantage of new technology without rewrite the whole system, this framework provides a refactoring solution for web sites built on CGI technique.

Keywords: XML, OOAD, Framework, Form, refactoring.

1. 前言

近年來，由於網際網路的普及化，以及它所帶來的便利性與商機，各個行業莫不將企業相關事業由傳統的作業方式改採網路作業模式，以趕上這股 e 化的潮流。然而，因為沒有了人工作業的處理，因此，對於使用者所提供的資料之正確性便必須由程式來負起檢驗的工作。對於資料的檢驗工作，一般來說可以從客戶端(client side)以及伺服器端(server side)這兩方面來考量。一般的商務網站，如果資料格式檢驗的工作是在客戶端執行，則大多均採用 JavaScript 來負責執行此部分的工作。如果資料格式檢驗的工作是在伺服器端執行，表單資料檢驗邏輯大多是與伺服器端所使用之程式語言寫在一起。這樣一來，隨著企業的處理邏輯更新，程式碼也必須跟著修改。這一點增加了程式的複雜度並且也降低了系統的可維護性。本研究是一實務性之論文，因此，如何設計一套軟體元件來簡化此繁雜的工作是本研究的重點之一。

另一方面，早期以 CGI 技術建置之網站，隨著軟體技術的進步，無法享受到其所帶來的優勢。基於有限的成本、時間、金錢與其他資源的限制，企業可能無法針對新的技術全面重新改寫系統。對此，本研究也嘗試提供一個有效的解決方案。

以下各節摘要如下。第 2 節介紹物件導向設計的優點以及本框架採用 XML 之相關技術，第 3 節討論本框架所提出之嵌入式表單的處理流程之基本架構，並且從客戶端以及伺服器端兩分面分析表單狀態，表單元件之物件化以及由 XML 所定義之表單定義檔結構。第 4 節說明系統架構並且使用 UML 視覺化語言表

達設計之類別庫，組成框架之類別圖以及動態循序圖，第 5 節說明系統實作部份並且提出應用範例來做使用說明。

2. 技術探討

2.1 物件導向技術

隨著資訊系統建置的複雜化，物件導向技術的應用以及開發方式漸漸取代傳統的結構化的系統開發方法。物件導向設計之優點在於其所提供之封裝(Encapsulation)，繼承(Inheritance)，以及多型(Polymorphism)等抽象概念。封裝的特性使得程式易於理解，繼承的特性使得程式得以再利用，而多型的特性讓程式易於將來的擴充以及維護。基於這些物件導向技術在軟體設計上所提供的優點，我們採用物件導向分析與設計(OOAD)的方法做為本研究之表單驗證框架的設計基礎。

2.2 XML 技術

近年來，由於電子商務的蓬勃發展，如何在各個異質平台上訂定一個一致的資料交換基礎變得迫切需要。而 XML 及相關技術適時地對此問題提供了一個解決方案。XML (eXtensible Markup Language) 可延伸性標示語言由 W3C 組織於 1998 年公佈，主要係以網際網路的作業環境為主，並具備了描述文件格式結構，資料再用及跨平台作業等特性[2]。XML 屬於一種標籤語言，其寫法十分類似 HTML，但 XML 並不是用來編排文件內容，而是用來描述資料，使用者需要自己定義描述資料所需的各種標籤。XML 的主要目的是提供文件資料的結構化[1]. 它的優點在於:

1. 具有良好格式以及驗證機制。
2. XML 是跨平台間資料交換的標準。
3. XML 是資料交換和 WEB 資料管理的重要技術。

而 XML 與 HTML 的差別在於:

1. HTML 目的是顯示資料，而 XML 的目的是描述資料。
2. HTML 有固定的語法和架構，而 XML 需依使用者需要定義文件架構。
3. HTML 標籤是預先定義好的，而 XML 需要自己定義所需標籤。

一般的瀏覽器只能讀懂 HTML 語言，而 XML 文件本身只定義資料內容，因此需要一種機制來描述 XML 元素如何被顯示，這種語言稱為樣式語言，XSL(eXtensible Style Language)是配合 XML 的樣式語言[5]。因此，除了做為資料交換的格式之外，XML 搭配 XSL Transformation 技術[8]就可以達到產生動態 HTML 網頁的效果。

綜合以上的描述，本研究採用 OOAD 以及 XML 等技術，提出建構一套植基於 XML 技術之電子商務表單驗證框架 (form validation framework)。此框架之優點在於使用者可以在不修改程式碼的情況下，根據檢驗需求，更改檢驗條件。這不僅可以提升程式的可維護性。並且，藉由框架設計的優點，也提供了該軟體元件在未來的擴充性(Extensibility)。

3. 系統分析與設計

3.1 處理流程架構圖

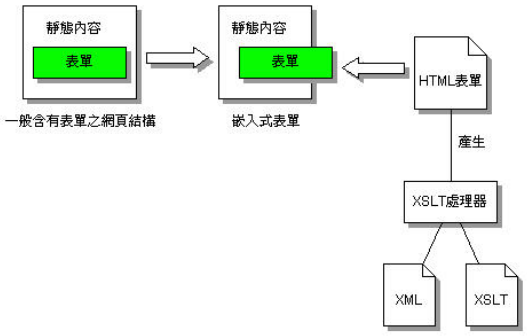


圖 1 處理流程基礎架構圖

本研究提出以嵌入的方式做為產生表單的方式。嵌入式表單的處理流程如圖 1 所示。圖 1 之左上角顯示一般含有表單之網頁結構。一般含有表單之網頁，依內容之性質，我們可以將其區分為兩大部分，一部分包含靜態內容，另一部份則是由表單元件所組成。圖 1 之中圖則顯示嵌入式表單結構。嵌入式表單基本上只是一個含有<FORM>標籤之 HTML 區塊，然而此 HTML 區塊必須經由 XML 與 XSLT 的轉換最後由 XSLT 處理器來產生以達到嵌入的效果。

3.2 客戶端分析

動態網頁技術的成熟發展，造就了電子商務系統的廣泛建置。這些技術包括了 Sun 所提倡的 Java Servlet，JSP 技術, Microsoft 的 ASP, ASP.NET 技術以及很多人廣泛採用的開放原

始碼 PHP 等等。但是不論是採用何種語言，最終呈現在使用者面前的還是由 HTML 所組成的網頁。HTML 是一種標記語言[7]，由許多的標籤 (tag) 所組成。為了能夠接受使用者的輸入資料，並且將資料傳送到伺服器端做處理，我們必須使用到 <FORM> 這個表單標籤以及其內含的一些表單輸入標籤。利用這些標籤，我們可以在網頁中做出各式各樣的輸入表格，讓使用者輸入適當的資料，然後透過 HTTP 通訊協定， 將資料傳送給伺服器端。

另外，在含有表單的網頁設計上，我們希望一旦有任何錯誤資料發生時，系統會將網頁導向至原先的輸入網頁，並且一併顯示錯誤訊息告知使用者，所以我們將表單組成部份視為一個整體，它所顯示的內容必須動態產生。因此，我們採用嵌入的方式載入 HTML 的表單。為了達到這個嵌入式的功能，我們可以使用 HTML 的內嵌框架標籤<IFRAME> [7]。使用 <IFRAME>將表單組成部份以及其他靜態網頁組成部份分開的優點是：可以讓網頁動態不會影響到靜態的顯示，另外，動態部份可以獨立測試直到沒有問題之後，再與靜態部份組合。

3.3 伺服器端分析

3.3.1 表單狀態分析

一般典型的電子商務網站均採用以表現層(presentation layer)，中介層(middle layer)，以及資料層(data layer)之 3 層式設計架構(three-tier architecture)。其中，中介層以處理商務邏輯為其主要工作。位於中介層之中央控制器負責處理使用者所送出之請求。而處理流程之主要工作可以簡述如下:

1. 剖析(Parse) HTML 表單元件資料。
2. 執行所需之檢驗工作。
3. 假如有任何錯誤，導向至原輸入網頁，並且顯示錯誤欄位資料。
4. 如果沒有任何錯誤，導向至成功網頁。

從使用者的觀點，一個含有表單的網頁會因為使用者所執行的動作而處於不同的狀態。例如：當使用者第一次進入該網頁，表單是處於一個全新的狀態。使用者可能會按下刷新(refresh)按鈕，這時表單是處於 refresh 狀態。如果使用者填完表單，按下提交(submit)按鈕，這時表單的狀態是處於 submit 狀態。最後，當使用者所輸入之資料不符時，我們必須將錯誤的訊息新增至原先的網頁上以告知使用者表單上有哪些錯誤。

從上述的分析，我們可以將一個含有表單的網頁可能處於的狀態分為下列四種：First Time, Refresh, Submit, Has New Content。其中，First Time 表示這是使用者第一次進入該網頁。Refresh 表示使用者按了 Refresh 按鈕。Submit 代表使用者已經輸入了表單資料並且按了提交按鈕。Has New Content 代表使用者所提交之表單有錯誤。對於所分析的結果，我們利用 UML[4]的狀態圖，表達如圖 2。

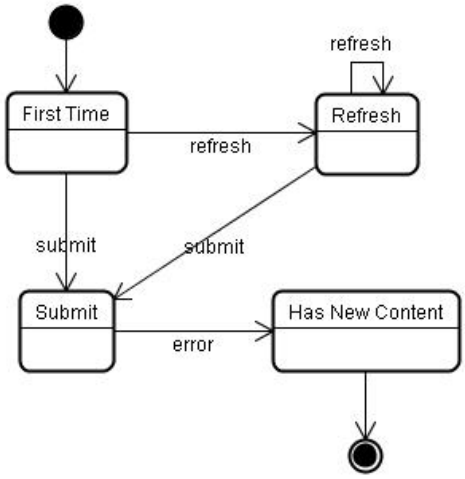


圖 2 狀態圖

3.3.2 表單狀態設計

在設計上，我們可以採用設計樣式中的 State 樣式(State design pattern) [3]來設計狀態物件。State 樣式的結構如圖 3 所示。採用狀態樣式的優點在於將各個不同狀態以物件的方式封裝成型，必免一堆條件判斷或是 switch 的敘述出現在程式中，這可使得程式中的邏輯變的更精簡且易於維護，另外，對於將來的功能擴充也更有效率。

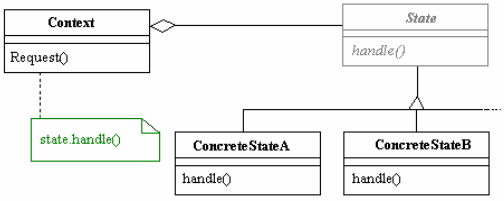


圖 3 State 設計樣式

在此處所謂的"狀態"是用以表達一個表單所具有狀態，而實際造成此狀態的來源來自

於使用者的行為。因此，在設計上，我們以一個 RuntimeData 物件來對應到 State design pattern 中的 Context 物件。 RuntimeData 類別封裝一個使用者在系統執行時的資料。圖 5 顯示根據 State design pattern 所設計之表單狀態類別圖。

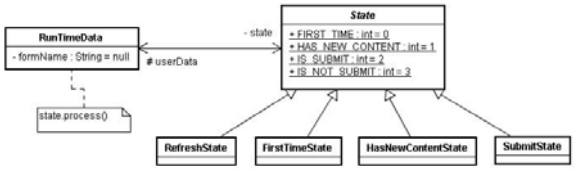


圖 4 表單狀態類別圖

3.4 表單定義檔設計以及 XML 物件

HTML 所使用的標籤由 W3C 所規範[6] 。在眾多標籤中，與表單相關之輸入元件由<INPUT>標籤元素來控制。因此，對於各種不同類型之表單輸入元件，我們以 XML 來為其定義所需之元素或是屬性。我們稱此 XML 檔案為表單定義檔(Form Definition File)。表單定義檔結構以及各元素、屬性之定義如表一所示：

表 1 表單定義檔

標籤名稱	屬性/解釋
<form>	name 屬性。名稱必須與定義檔同名。
<redirect>	包含表單之網頁的 URI 或是 URL。
<successful>	當表格成功提交之後的網頁 URL。
<fields>	<field>的父元素。
<field>	name 屬性：欄位名稱。必須與 HTML 元件之名稱相同。對於 checkbox 元件，命名格式為：ckBox_{group_tag}_{element_name}
	child 屬性: 是否含有子元件。
	sequence 屬性: 序號。
	required 屬性: true/false。
	message 屬性: 驗證錯誤時所要顯示的訊息。
	type 屬性: 表單元件之型態。例如：radio, textbox, select, checkbox 等。
	multiple 屬性: true/false。只有 SELECT 元件具有這個屬性。
	dataType 屬性: number / date。只有 textbox 或是 text 元件可以使用。
	dataTypeMessage 屬性: 當資料型態不符合時所需顯示的訊息。
	dateFormat 屬性: 只有當 dataType 是 date 時才可以使用。
	<match>
	<field>的子元素。使用於階層式之元件組。value 屬性:對應到 radio 元件之名稱。

4. 系統架構

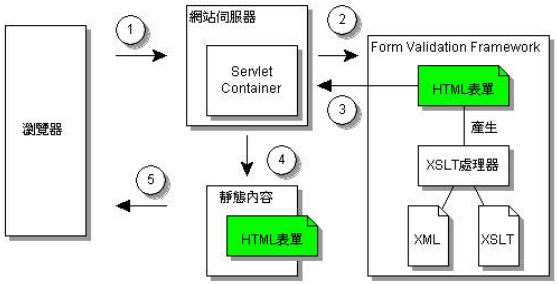


圖 5 系統架構圖

本研究採用 Java 語言實作文中所討論之框架設計。本框架主要由五個類別庫所組成，如圖 6 所示。其中，主要的類別庫及其功能為：

- Controller 類別庫：此類別庫負責處理任何流程之控制。在伺服器端，我們設計一個 Java Servlet 物件做為伺服器端之中央控制器，用以協調 HTML 表單的擷取與分析、使用者狀態之判別、表單驗證之執行與轉換以及網頁導向的工作。
- Definition 類別庫：此類別庫包含各項表單元件類別之定義。
- Mail 類別庫：在許多實務的設計上，當商務處理完成時，有需要寄發 email 來通知客戶這項功能。因此，以 Java Mail API 為基礎，此框架包含 email 模組負責執行此任務。

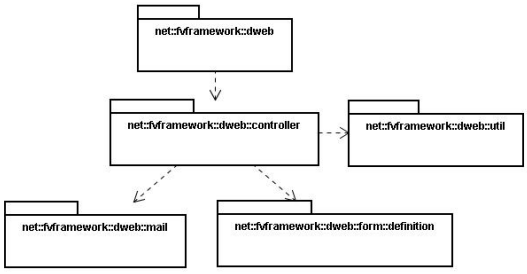


圖 6 類別庫相依圖

4.1 表單元件類別圖

圖 7 顯示表單元件定義類別圖之靜態結構(類別之操作省略)。

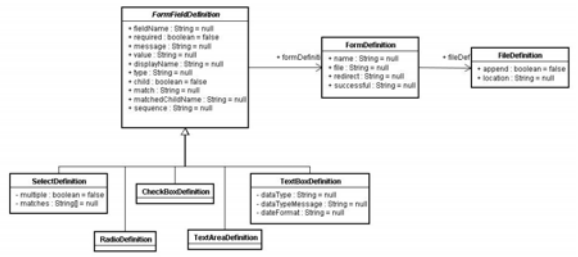


圖 7 表單元件類別圖

4.2 循序圖

對應於每個表單，此框架設計一個負責該表單之 Servlet。當 Servlet 接收到使用者的請求時，該 Servlet 會讀取相關之表單定義檔，並載入該檔案中所定義之表單元件規則，如圖 8 所示。

接著，控制器會判斷使用者目前所處於的狀態，然後決定所要使用到的狀態物件，最後藉由呼叫該狀態物件之 process()方法，將控制權交由該狀態物件來處理，如圖 9 所示。

真正的表單檢驗工作只有當使用者提交了表單之後才會執行。此時之狀態為提交狀態(SubmitState)。圖 10 顯示當使用者處於提交狀態時，表單驗證框架如何呼叫各相關物件以從事表單元件之驗證工作。每一個表單元件定義(例如 TextBoxDefinition) 類別在表單初始化時已經根據設計者所提供之表單定義檔之內容載入。由圖中我們可以看出，對於任意之表單，只要根據網頁表單之需求設計好表單定義 XML 檔，框架本身會依照設計自動地呼叫各類別之 validate()方法。

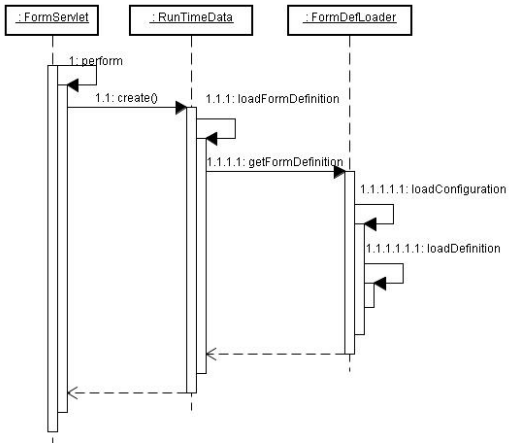


圖 8 控制器初始化循序圖

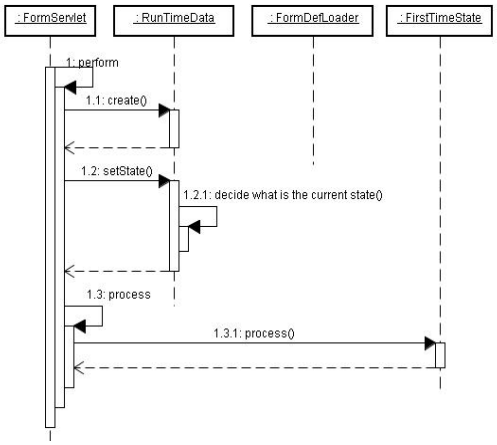


圖 9 呼叫狀態圖

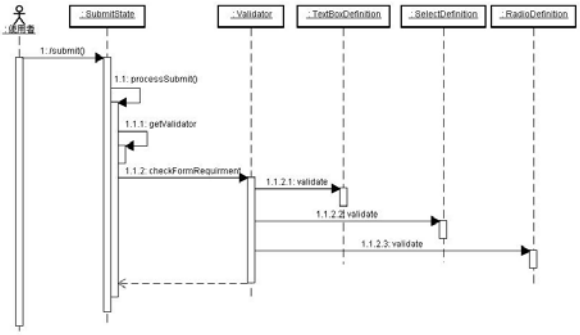


圖 10 提交循序圖

5. 系統實作

5.1 客戶端設定

依據處理流程基礎架構圖，客戶端的設定非常簡單，假設我們有一個網頁 OrderWater.html 必須包含一個表單，那麼，我們使用如下的程式碼來將其嵌入於網頁中：

```
<IFRAME id="formFrame"
src="/fvframework/FormController?form=Water
OrderForm" width="500" height="2000"
frameborder="0" Scrolling="No">
</IFRAME>
```

其中，fvframework 對應到網路應用程式的根目錄，FormController 是一個伺服器端的控制物件名稱，其後描述對應的表單定義檔案名稱。

5.2 表單元件 XML 設定

本研究針對 Select，Radio，Checkbox，

Textbox， TextArea 等五個不同類型之表單輸入元件給出定義。下面舉例說明如何在表單定義檔中描述所需之 XML 以及規則：

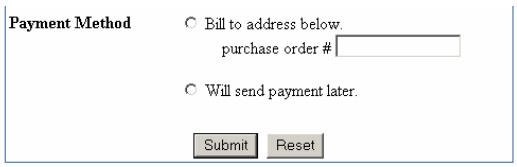
1. Textbox/TextArea 元件之設定: 假設有一個 Textbox 元件，其規則是值必須非空。在設定上，可以採用以下的方式：

```
<field name='name'
        sequence="1"
        displayName="User Name"
        required='true'
        type="text"
        message="Name is required."/>
```

2. 日期格式之設定：假設表單要求我們依照某特定的格式輸入日期，為了確保此規則的要求，在設定上，可以採用以下的方式：

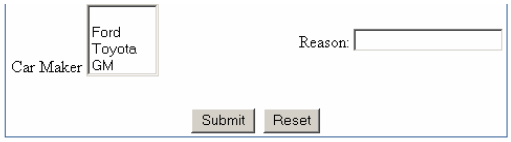
```
<field name='birthday'
        sequence="9"
        displayName="Birthday"
        required='true'
        type="textbox"
        message="Birthday is required."
        dataType="date"
        dateFormat="yyyy/dd/MM"
        dataTypeMessage="Birthday should be in
        this format yyyy/dd/MM"/>
```

3. 具有關聯性元件之設定：假設我們有一個如下的表單，其規則是當使用者點選”Bill to address below”時，purchase order #這個欄位不可以是空白。為了確保此規則的要求，在設定上，可以採用以下的方式：




```
<field name='payment'
        sequence="10"
        displayName="Payment Method"
        type="radio"
        required='true'
        message="Please select payment method.">
        <match value="Bill to Address Below">
            <childfield>poNumber</childfield>
        </match>
</field>
<field name='poNumber'
      child='true'
      sequence="11"
      displayName="Purchase order number"
      required='true'
      type="text"
      message="Purchase order number is
required."/>
```

4. 多值關聯性元件之設定：假設我們有一個如下的表單，其規則是當使用者選擇(多選)Car Maker 中的 GM 或是 Ford，我們要求使用者必須提供 Reason。為了確保此規則的要求，在設定上，可以採用以下的方式：(此範例與範例 3 之差異在於使用不同類型之表單元件)




```
<field name='maker'
        sequence="12"
        displayName="Car Maker"
        required='false'
        type="select"
        multiple="true"
        message="Please select a car maker.">
        <match value="Ford,GM">
            <childfield>reason</childfield>
        </match>
</field>
<field name='reason'
            child='true'
        sequence="13"
        displayName="Reason"
        required='true'
        type="text"
        message="Reason is require."/>
```

6. 結論

本論文提出一套採用物件導向設計之電子商務表單檢驗框架，所具有之特點可以歸納如下：

1. 採用物件導向設計將表單元件物件化，並且加入驗證邏輯規則，對於將來有關表單之新增元件可以容易地擴充。
2. 在不需要修改程式碼的情況下，可以容易地更改錯誤訊息。
3. 此框架可以獨立存在，並且被嵌入於任何之電子商務網站。
4. 與網頁靜態內容做切割，讓系統易於維護性。
5. 本框架設計並不依賴於特定領域，使得此

框架有極大的再利用性。

本文中所提出之表單驗證框架可以應用到任何須要使用到表單之電子商務網站，尤其是對於採用早期 CGI 技術建置之網站，在有限的成本、時間、金錢與其他資源的考量下或不需要全面重新改寫系統，但又希望能夠運用新的技術所帶來的優點的話，本研究提供了一個有效的重構解決方案。

本研究所開發之程式碼可於此處下載: https://sourceforge.net/projects/fvframework/

參考文獻

[1] 梁中平、徐子淵、謝鎮澤，***XML 與電子商務標準***，財團法人資訊工業策進會，2000。

[2] 梁中平、蔡峻雄，***電子商務導航***，資策會電子商務應用推廣中心，第五卷，第五期，2003。

[3] Gamma E., Helm R., Johnson R., Vlissides J., ***Design Patterns: Elements of Reusable Object-Oriented Software***, Addison-Wesley, 1995.

[4] Booch G., Rumbaugh J., Jacobson I., ***The Unified Modeling Language User Guide***. Addison-Wesley, 1999.

[5] World Wide Web Consortium (W3C), ***The Extensible Stylesheet Language Family*** (XSL), http://www.w3.org/Style/XSL.

[6] World Wide Web Consortium (W3C), ***HyperText Markup Language*** (HTML) Home Page, http://www.w3.org/MarkUp.

[7] ***World Wide Web Consortium*** (W3C), http://www.w3c.org.

[8] J. Clark, ***XSL Transformations*** (XSLT) version 1.0, 1999.

Appendix: sample Form Definition File

Sample Form Definition File

```
<dweb>
<form name="WaterOrderForm">
<srcfile>formWater.xsl</srcfile>
<redirect>/forms/OrderWater.html</redirect>
<successful>OrderWaterSuccess.html</successful>
<!-- mail tag is not required -->
<mail send="true">
        <subject>Order Water Request</subject>
        <from>fsyu@mail.ksu.edu.tw</from>
        <to>fsyu@mail.ksu.edu.tw</to>
        <!-- <cc>fsyu@mail.ksu.edu.tw</cc>      -->
        <!-- attachment tag is not required, and can be multiple
        <attachment location="c:\\test.pdf"
        displayName="test.pdf"
        />
        -->
</mail>
<!-- file tag is not required -->
<file append='true'>
<location>c:\orderWater.txt</location>
</file>
<fields>
        <field name='name'
                sequence="1"
                displayName="User Name"
                required='true'
                type="text"
                message="Name is required."/>
        <field name='itemized'
                sequence="3"
                displayName="Itemized Price Quote"
                type="radio"
                required='true'
                message="Please select Itemized Price Quote."/>
                <!-- For check box, naming convention could be
                ckBox_{group name}_{element name}
                -->
        <field name='ckBox_TR_coliform'
                sequence="4"
                displayName="Tests Requested: Total coliform bacteria"
                type="checkbox"
                required='true'
                message="Tests Requested is required."/>
        <field name='ckBox_TR_nitrate'
                sequence="5"
                displayName="Tests Requested: Nitrate "
                type="checkbox"
                required='true'
                message="Tests Requested is required."/>

        <field name='pizzasize'
                sequence="7"
                displayName="Pizza Size"
                required='true'
                type="select"
                multiple="true"
                message="Please select one pizza size."/>
        <field name='age'
                sequence="8"
                displayName="Age"
                required='true'
                type="textbox"
                message="Age is required."
                dataType="number"
                dataTypeMessage="Age should be a number"/>
        <field name='birthday'
```

```
            sequence="9"
            displayName="Birthday"
            required='true'
            type="textbox"
            message="Birthday is required."
            dataType="date"
            dateFormat="yyyy/dd/MM"
            dataTypeMessage="Birthday should be in
this format yyyy/dd/MM"/>

        <field name='payment'
            sequence="10"
            displayName="Payment Method"
            type="radio"
            required='true'
            message="Please select payment
method.">
            <match value="Bill to Address Below">
            <childfield>poNumber</childfield>
            </match>
        </field>
        <field name='poNumber'
            child='true'
            sequence="11"
            displayName="Purchase order number"
            required='true'
            type="text"
            message="Purchase order number is
required."/>
</fields>
</form>
</dweb>
```