關貿第 16 期電子報

封面故事：E 化無境界，關貿導入 SSO 讓用戶便利又快速

在流通行業 e 化挹注許多的關貿網路，為了讓用戶更便利使用各電子資訊系統，公司研發多年，終於導入 Single Sign On（SSO）技術，未來用戶只需要一組帳號跟密碼，就可通行無阻，服務全新升級，要讓客戶更加滿意。

前身為「財政部貨物通關自動化推行小組」（簡稱通關小組）的關貿網路，於 1996 年與民間企業合資成立，從一開始負責政府通關自動化業務，建構第一個國家級 EDI 通關資訊交換網路，至協助企業供應鏈電子化、採用電子商務，已成為公司要 e 化的首選優質服務商代表。

只需一組通關密碼就能暢通無阻

以電子化服務見長的關貿網路，向來將企業電子化服務置於第一優先，公司已開發出超過五十套不同的 ASP 系統，而客戶通常也不會只有使用一套產品，可能常用的就有數十個系統，不過過去用戶的經驗是，只要登入一個系統就需要一套帳號跟密碼，因此登入數十個系統就需要數十個帳號跟密碼。

但現在關貿網路已經將系統重新升級，導入 Single Sign On（SSO）技術，也就是相同權限的使用者都只需要登入一次帳號與密碼，大幅度降低了用戶的困擾，在分秒必爭的這個時代，「快速、有效」是每個用戶的需求，而使用過的用戶也確實覺得相當便利，同時也覺得這算是相當大的突破創新。

關貿網路表示，大約在 4~5 年開始就有這個想法，不過因為整體系統要升級，需要龐大的技術工程，而研究過市場上許多產品，加上主事者的堅持，儘管困難重重，但逐一更新所有系統，最後終於在今年完成服務升級，這也顯示關貿網路追求服務要盡善盡美。

E 化的下一步整合成商業智慧

在內需市場愈來愈熱門的流通行業，目前有許多大廠都是關貿網路的客戶，包括家樂福、全聯社、屈臣氏、康是美、頂好 Welcome、松青，而正因為現階段消費者行為很複雜，業者也希望能抓住消費者真正的心，因此透過系統的真實紀錄，再經過客製化報表與分析模型，就能讓流通業者更清楚了解市場的反應。

關貿網路表示，這種稱作「商業智慧 BI」的服務，也是明年度推廣重點之一，而這些資訊情報的蒐集不僅僅是數字的顯示，而是經過統計分析與整理，就能讓用戶了解數據所產生的背後意義，解讀目前現

象與未來消費流行的趨勢。

關貿網路舉例,目前許多流通業者都會發送 DM 或者是 DM 上附有折價券訊息，每次發放時可能都要花上 200~300 萬元，一年下來也需要上千萬元，不過這些 DM 折價券的回應率可能不高，有時候不到 2%，但透過商業智慧系統的協助，不僅能更精準的抓住客戶的心，提高回應率，還能維繫顧客關係，也能幫業者帶來實質的收益。

而在景氣歹歹的時代,供應商業者都希望能拓展商機,相對的流通業者也想要找尋更多的流行新品，為了滿足雙方的需求，關貿網路與條碼促進會攜手合作，採用該會 GDSN 國際標準，建置商品資料庫，方便讓世界各國的流通業者尋找商品,尤其在兩岸互動益加頻繁的現代，對岸的量販、或者超市也可以透過此系統來台找商品。

關貿網路提到，目前旗下的客戶，家樂福內部已經率先在推動此系統，以家樂福全球第二大量販店的規模來看，供應商加入後，其特色商品也有更多機會鋪貨至其他國家，所謂的「貨暢其流」的境界應該就是如此。

新聞專區：關貿網路陳振楠總經理當選亞太電子商務聯盟新任主席

亞太電子商務聯盟 (Pan Asian E-Commerce Alliance，簡稱 PAA) 第三十屆指導委員會議已於 2008 年 12 月 4-5 日在香港舉行。

指導委員會表示在過去十二個月中 PAA 跨國傳輸量不斷成長。同時因擁有建置完善之基礎架構，亞太電子商務聯盟更致力發展跨國電子產證服務，其中由台灣與韓國合作之先導專案亦成為其他會員國之典範。

亞太電子商務聯盟指導委員會選出台灣關貿網路陳振楠總經理出任下屆主席，並由新加坡勁升邏輯有限公司總經理 Leong Peng Kiong 先生出任副主席。

非洲電子商務聯盟代表 Mouhamed Diouf 先生表示希望有機會與亞太電子商務聯盟合作，同時希望可創造非洲與亞洲之間跨境安全交易之合作契機。

「非洲電子商務聯盟的組成與主要是希望藉學習亞太電子商務聯盟各會員國之間互相合作之成功模式。亞太電子商務聯盟期待與非洲電子商務聯盟未來之合作機會，並互為觀察員，分享經驗和知識。」現任副主席香港 Justin Yue 先生說。

指導委員會表示非常感謝前任主席馬來西亞 Amiruddin Abdul Aziz 先生對於亞太電子商務聯盟過去二年來的領導與貢獻。

第三十一屆亞太電子商務聯盟高峰會會議將於 2009 年 4 月於台灣召開。

香港 SAR

2008 年 12 月 5 日

關於亞太電子商務聯盟

亞太電子商務聯盟成立於 2000 年 7 月，由亞洲各國結盟而成，主要以推展安全可靠之 IT 基礎建設與推廣高效能之全球運籌服務為主。結合各會員國家之客戶，估計約有 155,000 家企業已幾乎包含所有亞洲市場之貿易商。目前亞太電子商務聯盟會員國包含泰國 CAT Telecom Public Co. Ltd、中國大陸中國國際電子商務中心 (CIECC) 、新加坡勁升邏輯有限公司(CrimsonLogic)、馬來西亞 Dagang Net、韓國貿易網路(KTNet) 、日本 NACCS 中心、澳門 TEDMEV、香港貿易通電子貿易有限公司(Tradelink) 、台灣關貿網路股份有限公司(Trade-Van)、日本 TEDI Club、澳洲 Tradegate 及菲律賓 InterCommerce。

如欲取得更多亞太電子商務聯盟資訊，請至 www.paa.net.


泛亞電子商貿聯盟第三十次督導委員會會議
The 30th PAA Steering Committee Meeting

服務新訊：【您的資料備份有用嗎？淺談異地備份回存之重要性】

文/關貿網路 陳凱勝

凡事不怕一萬，只怕萬一，異地備份回存驗證也決非杞人憂天，凡事以最壞的打算、來做最齊全的準備，就不用怕有萬一狀況發生時，資料卻沒有備份齊全，損失的不只是有形的客戶資料與金錢，更有難以彌補的商業信譽問題。

隨著資訊科技的進步，資料已成為現在企業最寶貴的資產，甚至是企業的命脈。以前總認為磁帶備份（Backup）等於是異地備援，但是，當備份完成了是否有想過其備份磁帶可以真的回復（Recovery）資料與系統？是否注意到資料的可用性（Available）？因此在分秒必爭的時代中，如何保護資料不遺失、損壞及災難發生後的迅速復原，並繼續維持正常的運作，已成為現在企業確保業務執行不間斷所必須進行的基礎工程。諸如美國911事件，讓資料雙備援系統(HA)同時建置在雙子星大樓的企業，異地備份資料因突如意外，資料頓時毀於一旦；台灣汐止東帝士大樓大火，也讓受災企業的資料全數付之一炬，MIS只能重頭再來；而橫掃美國田納西州的強烈颶風，也讓許多企業的資料「隨風而逝」。

「知而言不如起而行，莫讓意外發生時，才體會到異地備份回存

驗證的必要性。」

異地備份回存驗證不是新趨勢，只是因為一味地強調資料應如何備份才是最完整解決方案，卻忽略備份真正的最終目地是『 **資料要能夠完整回復且可運用** 』。

目前有建置異地備份回存驗證演練的企業少之又少，其中原因不外乎許多企業都覺得倒楣的一定不會是我，有些單位覺得只要本地端有作磁帶備份或是雙備援機制就已足夠，總不至於整個本地端都完全毀損吧！但意外就是所始料不及的，當資料毀損或無法完整回復，則就真的是叫天天不應，叫地地不靈，因此，當企業遭遇到不可抗拒、無法預期的天災人禍，若平時有做好企業系統與資料的備份回存驗證演練，絕對會是應付緊急狀況的最佳解法。

為了更好的分散風險，我們不但要把雞蛋放在不同的籃子裡，最佳方式更要把它們分散至不同的地方。企業一般在進行備份的時候常常只注意到資料備份（Data Backup），而忽略了系統備份（System Backup），其實無論是系統或資料都相當重要，對某些產業而言，系統毀了無法立即提供服務，所造成的損失將是無法估算。

關貿網路公司技術顧問團隊在多次協助客戶驗證經驗中，察覺大部分客戶對於備份遭遇到以下問題：

1. 公司雖已建置 HA 雙主機備援架構，但天災發生時，往往同一地區的主機都無法倖免於難。且遇到這種狀況，緊急調度的機器，規格往往與線上主機大不相同，且更難以確保磁帶資料是否可以完整回存 ？

2. 一般都單純驗證資料庫資料回存，但若要異機回存，系統檔案架構是否完整備份 ？ 應用程式及資料庫 Kernel Source 是否有考慮到 ？

3. 若線上主機使用已數年，面臨作業系統版本老舊，而異機的作業系統版本是最新版本，如何確認資料庫備份與新作業系統的匹配性 ？

為了解決企業客戶所面臨的挑戰，並模擬企業遭逢"萬一"狀況，技術顧問團隊特別規劃『異地備份回存驗證服務』，在不影響現用資料及企業正常運作下，為您：

- 驗證 ERP 主機備份完整性 (含 OS、AP、DB)。
- 災難預防備援演練(含 ERP 備援主機重建演練及異地備援步驟演練)。
- 提供完整完整測試報告及改善建議，最專業的技術團隊隨時支援，即時地解決您的問題，

「多一分準備，則多一分安心」。

異地備份回存的驗證服務流程

異地備份回存驗證的效益：

1. 驗證平時 ERP 主機備份機制之完整性(含 OS、AP、DB)。
2. 正確掌握 ERP 系統復原所需的資源與時間。
3. 提供基本備援機制。
4. 意外災害後能快速將備份磁帶還原。
5. 加強企業災難緊急復原演練。
6. 防災演練及早發現並修正缺失防範未然。

相關資訊安全服務請洽詢： 安全、品質、服務、效率
關貿網路股份有限公司 陳凱勝
TEL：02-26551188#517
e_mail：shenghung.chen@tradevan.com.tw

資安小常識：深入淺出應用程式資料驗證議題

關貿網路公司李泳泉

為何要作資料驗證?

孫子兵法軍形篇說道：「昔之善戰者，先為不可勝，以待敵之可勝」；在戰場上，要先使自己不可被戰勝，才有機會戰勝別人。同樣的，系統也必須要能鞏固自己的防禦，才能在惡意使用者的圍剿下生存；而資料驗證（Data Validation）正是鞏固系統防禦最重要的方法。

「所有的輸入都是惡意的（All Input is Evil!）」是微軟的安全專家在 Writing Secure Code[1]這本書所提出的至理名言，也一語道出資料驗證的必要性；然而由近幾年來 Cross-Site Scripting[4]及 Injection Flaw[5]等注入式弱點排名在 OWASP Top 10 中不降反升的趨勢可以看出，資料驗證在許多系統中沒有被充份落實。

資料驗證是個很一般化的詞，相信大多數開發人員在接收到「系統要作輸入驗證」這樣的指令時都會不知所措。因此在本文中，我們嘗試將資料驗證的相關議題作整理、劃分種類、並在最後提出可行的實作建議。

資料驗證的種類

資料驗證的目的是檢驗資料的合法性；對系統而言，資料合法性不外乎以下三個：(1)資料一致性、(2)是否包含惡意的字元、以及(3)是否合乎特定特徵或商業邏輯。

(1) 資料一致性（Integrity）

驗證資料在傳輸過程是否遭到竄改？對於某些使用 Cookie 或 HTTP Header 作為通過認證依據的單一簽入機制，若 Cookie 或 Header 內容遭到竄改可能造成惡意使用者偽裝其使用者或不當的提升權限。

(2) 資料中是否包含惡意的字元

包含 HTML 或 Script 的資料內容（如…<a onClick="…">…）可能造成 Cross-Site Scripting 攻擊；而包含單引號（'）或兩個連字符號（--）的輸入可能造成 SQL Injection 效果；除非系統有特殊需求，一般而言資料內容不應包含這些字元。

(3) 資料是否符合特定特徵或格式

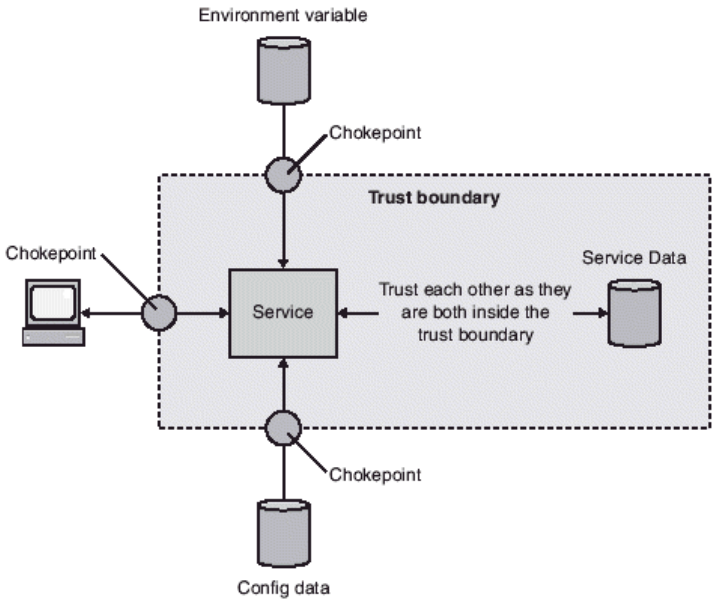
大多數的系統資料都可以被規範出其應具有的資料特徵（例如型別、長度、範圍等）或是應符合的格式（例如 Email 格式、中華民國身份證字號格式等）；系統應該要可以識別資料是否符合該特徵，以決定是否使用該資料。

不同的合法性有不同的驗證方式；在實作資料驗證時我們必須明確的知道該資料所應俱備的合法性，才能夠選擇適當的驗證邏輯及實作方法；例如身份證字號的輸入資料應該被驗證是否符合身份證字號格式。實作方法部份在稍後的章節會有較詳細的描述。

資料驗證的範圍及界限

為增加可維護性及修改的彈性，現在的應用系統大多採取多層式（Multi-tier）設計；但是要在何處實作資料驗證卻也成為許多系統設計師難解的問題。最理想的方法是每個層（Tier）都要實作資料驗證，但是耗費的工時、以及對功能和效能可能造成的影響讓理想的方法永遠只是理想。

Writing Secure Code[1]建議我們應該定義信任邊界（Trust Boundary），在邊界內的資料都是可以信任的。邊界的範圍依情況有所不同，但我們建議邊界的最大界限不應跨越系統，如圖表 1 所示（此圖亦出自 Writing Secure Code[1]）：



圖表 1 系統的信任邊界[1]

應用系統內的各層可以信任彼此提供的資料；但是一旦跨越邊界的檢核點（Checkpoint），資料就應該被驗證。

常見會跨越檢核點（Checkpoint）的資料如下：

- HTTP Request
  - Cookie
  - Header
  - POST Data (Form Data、Hidden Field、File Upload)
  - GET Data (Query String)
- 透過 Remote 技術呼叫或取得的資料
  - EJB
  - RMI
  - .NET Remoting
  - Web Service
  - Socket
  - RSS
- 系統之間的資料轉檔
  - File
  - Database

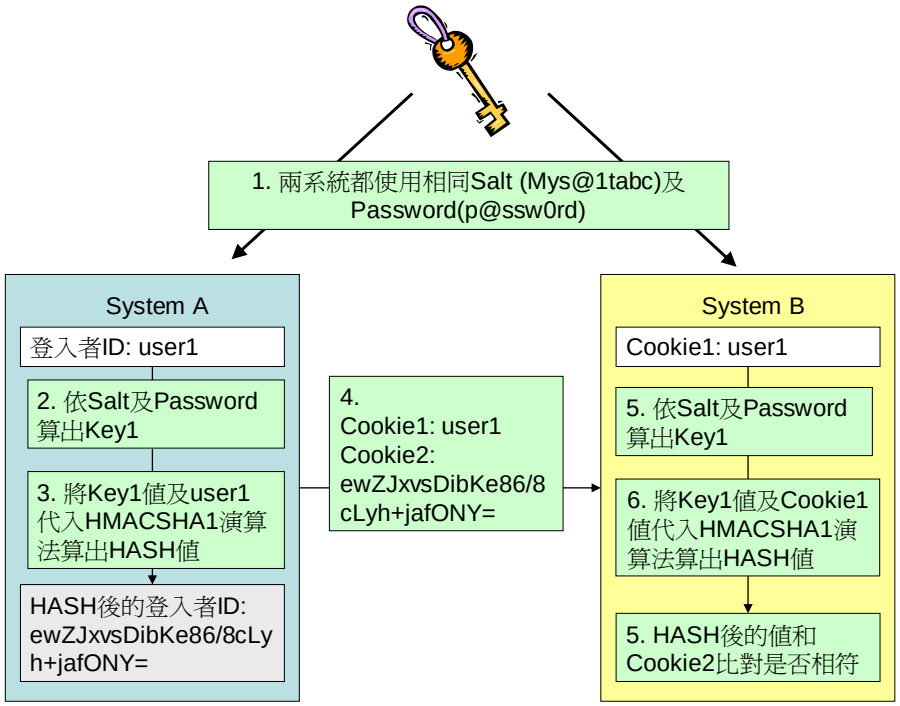
資料驗證的方法

在本小節裡，我們介紹資料驗證常用的方法；並與先前提到的三個資料的合法性連結，說明其適用性。

(1) 使用雜湊（Hash）演算法驗證資料完整性(integrity)

驗證資料一致性的目的在於確認資料在傳輸過程中完整性未遭非人為(accidental)破壞，和原始的值相同。使用雜湊（Hash）演算法是確保資料完整性(integrity)的常用作法。舉例來說，兩系統之間使用 Cookie 傳送認證資料（例如登入者的 ID），一個 Cookie（Cookie1）以明文方式存放登入者 ID，另一個 Cookie（Cookie2）則存放使用雜湊演算法運算過的登入者 ID；另一個系統接收到兩組 Cookie 後，先將 Cookie1 的內容用同樣的雜湊演算法運算後，再與 Cookie2 的內容比較，若兩者一致則代表傳輸過程中未遭非人為破壞。目前軟體發佈時常用 MD5 作為雜湊演算法；但 MD5 已被證明可找到雜湊碰撞(Hash Collision) [6]，亦即已被破解，因此在演算法選擇上建議使用強度較高的雜湊演算法（如 SHA256），以增加破解難度。如果要偵測資料在傳輸過程中，遭到人為蓄意(intentional)的篡改，則必須使用包含金鑰運算的雜湊演算法(Keyed Hash)，如 HMACSHA1。若使用

Keyed Hash 演算法，在運算之前雙方系統要先交換金鑰，或據以產生金鑰的參數，如圖表 2 中的 Salt(亂數值)及 Password。圖表 2 提供使用 Keyed Hash 演算法的運作流程示意。



圖表 2 使用 key 的雜湊演算法運作流程

(2) 使用黑名單阻擋不合法的資料輸入

所謂黑名單的作法是以思考”我的系統不能接受那些字元或字元組合的輸入”為出發點來進行資料驗證。黑名單的實作可以分為兩大類：（1）檢查並封鎖（2）消毒（Sanitize）

- **檢查及封鎖**

檢查資料內容是否包含不合法的字元或字串，若發現即拒

絕該資料進入系統；最簡單的方式是使用 String 類別內建的方法(如 Contains)來檢查資料中是否包含不合法字串，若要檢查的字串樣式（Pattern）較複雜；例如要檢查是否包含以<開頭，以>結尾的字串；則需要使用正規表示式[3]（Regular Expression）來進行檢查。ASP.NET 本身所具備的驗證 Request 內容的功能就是屬於檢查及封鎖的類型。

- **消毒（Sanitize）**

  不對資料進行檢查，而是直接對資料進行消毒；也就是使用取代（Replace）或編碼（Encode）的方式將有害的字元過濾掉；例如將上引號（’）取代為兩個上引號（’’）來阻止 SQL Injection，或是使用 Server.HtmlEncode 將小於符號（<）取代成<來防止 Cross-Site Scripting 攻擊的發生。

(3) 使用白名單列舉系統可接受的資料特徵

相對於黑名單，白名單的作法是以思考”我的系統只能接受符合特定特徵的輸入”為出發點來進行資料驗證。

資料特徵包含型別(文字或數字)、最大或最小長度、範圍(若為數字或日期）、以及特定格式（Email 格式、身份證字號格

式）。

驗證資料特徵最簡易的方法是利用 Framework 本身提供的函式來實作型別或長度的檢查；例如.NET 的基本資料型別，包括 Int16、Single 等類別都有提供 TryParse 方法，Java 則可透過基本資料型別的外覆（Wrapper）類別所提供的 ParseInt、ParseFloat 等方法來實作。

若要檢查的資料特徵較複雜；例如要檢查是否符合 Email 格式；則需要使用正規表示式[3]來進行檢查。

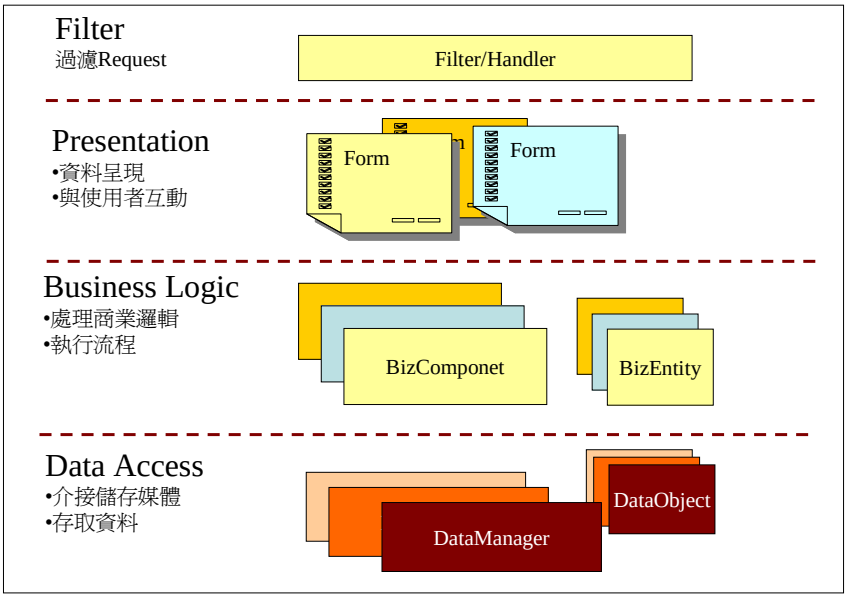
在驗證合法性的實作上，雜湊演算法適合用來驗證資料的一致性；黑名單的作法較適合驗證資料中是否包含不合法的字元；白名單的作法則適合資料特徵或資料邏輯的驗證。在實務上，黑名單的作法比較容易產生漏洞，因為資料的樣式可以有許多變化，例如 C14N 的議題[7]、URL Encode；黑名單的檢查清單可能無法包含所有的變化而無法阻擋惡意輸入。因此我們建議系統應搭配黑名單與白名單作法，讓惡意使用者難以趁虛而入。

資料驗證實作建議

在看過資料驗證的種類、驗證的界限、以及作法以後，是否還是不知要如何在系統裡實作資料驗證呢？前面介紹的就像是

一塊塊的拼圖，在本小節我們將會拿出拼圖板 - 系統架構；並將那一塊塊的拼圖放到適合的位置，幫助大家了解實作資料驗證後的系統全貌。

在前面曾提過，現在的系統為了維護及擴充的彈性大多採用多層（Multi-tier）式的架構設計，如圖表 3 所示：



圖表 3 多層式系統架構

大部份系統大致都可分為呈現層（Presentation Tier）、商業邏輯層（Business Logic Tier）和資料存取層（Data Access Tier）；.NET 和 J2EE 架構還有支援 Filter（.NET 為 HttpHandler；J2EE 為 Filter Servlet）的實作，允許程式在使用者的 Request 進入系統之前進行某些處理（例如編碼轉換）。由於每一層的責任及功能不同、資料特性也有差異，因此在每一層的資料驗證策

略也有所不同。接下來我們來檢視各層的責任及資料特性，以及建議適合的策略及實作方式。

在進行檢視之前先說明一個概念;注入式攻擊,如 Cross-Site Scripting 或 SQL Injection 之所以能成功是由於系統將資料（Data）當成指令（Command）的一部份來執行；資料驗證是針對資料進行，而非指令。因此在各層中，資料和指令是否能被區分也是資料驗證的一個重要考量。

- **Filter**

  Filter 會在所有 Request 進入系統前先對資料進行處理；雖然在 Filter 中資料和指令的區別是很明確的，但是在 Filter 一般不會實作商業邏輯，所以 Filter 較適合實作資料一致性的驗證、以及使用黑名單過濾 Request 中的不合法字元。若是在 Filter 中實作資料邏輯驗證，可能讓 Filter 的程式過於複雜而造成效能問題,若是因驗證規則錯誤也可能影響系統的全部功能。

- **呈現層（Presentation Tier）**

  呈現層是和使用者互動的第一線；可以清楚的區分資料和指令，同時也已清楚資料特徵及格式，因此適合使用白名單實作資料特徵及格式的驗證。若是架構不支援 Filter,使用黑名單方式過濾不合法字元的功能也可能在此實作。在呈現層實

作資料驗證的缺點是驗證程式會分散在所有頁面，實作和管理較費時；同時也需要處理錯誤的資訊呈現。因此在實作上建議使用 Framework 提供的既有元件，例如 ASP.NET 的 Validation Control 或 Java Struts 的 Validation 模組。

- **商業邏輯層（Business Logic Tier）**

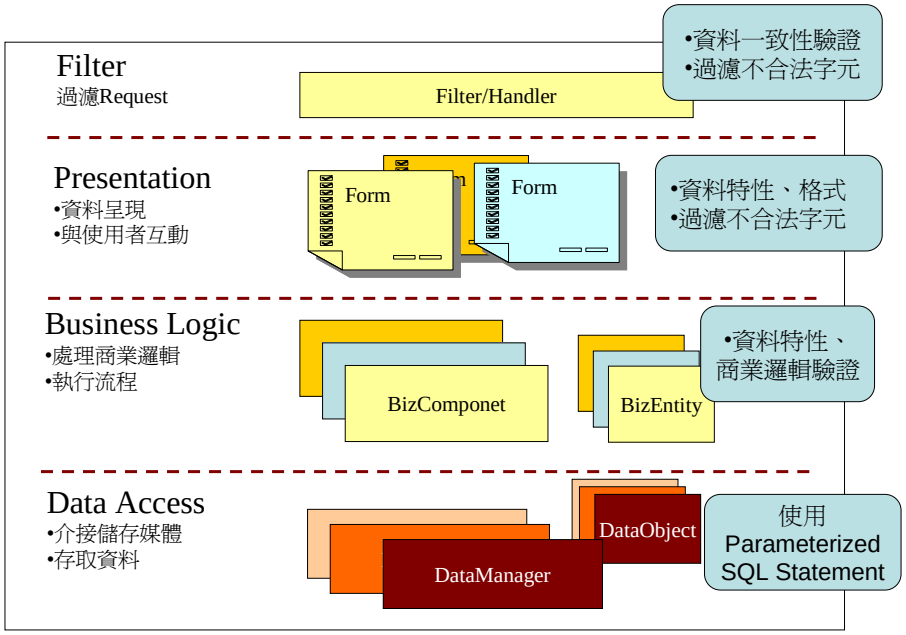
  商業邏輯層負責處理系統中商業流程的執行，在此亦可清楚區分資料和指令，同時也應該清楚資料特徵及商業邏輯；因此除了呈現層之外，商業邏輯層亦適合實作資料特徵及商業邏輯的驗證。由於不需要處理錯誤的資訊呈現，因此可以使用程式語言提供的方法（如 TryParse、ParseInt）或使用正規表示式實作驗證規則。

- **資料存取層（Data Access Tier）**

  資料存取層負責介接外部儲存媒體，進行資料存取。許多人會認為在此實作資料驗證是最輕鬆的，因為所有資料幾乎都會經過這裡；把守最後的關卡即可。但是從職責來看，資料存取層不應實作過多商業邏輯，若要檢查資料的特性，應該檢查的也只是資料欄位是否可以空白、型態和長度。而且有的實作是在商業邏輯層將 SQL 字串組好，再交由資料存取層執行；因此有可能在資料存取層已經無法區分資料和指令，

所以資料存取層並不適合實作資料驗證。在資料存取層能作的便是使用參數式的資料庫查詢指令（Parameterized SQL Statement）讓資料庫知道那些是資料、那些是指令，以降低 SQL Injection 發生的機率。

綜整以上的描述，我們可以得到一張新的架構圖，如圖表 4：



圖表 4 加入驗證的多層式系統架構

實作案例

最後我們以一個實作案例，輔以 ASP.NET 程式碼範例來實現上面所談的實作策略。我們希望為公司實作一個人力登錄系統，掛在公司入口網站下。求職者可以上系統留下自己的聯絡方

式，包括身份證字號、姓名、電話、住址、Email 等、個人履歷、以及感興趣的職缺，求職者在登錄資料時都不允許使用 HTML 及 JavaScript 以避免 Cross Site Scripting 攻擊；管理者需先由公司入口網站登入，才能管理求職者登錄的資料；人力登錄系統和入口網站是不同的系統，管理者在入口網站登入後，入口網站會以 Cookie 記錄登入者的帳號，讓人力登錄系統可以識別管理者是否已登入系統。

由於兩系統間會有 Cookie 的交換，為避免 Cookie 在過程中被篡改，因此我們在 Filter 中驗證 Cookie 內容的一致性，並且先檢查輸入的參數是否包含不合法的字元。以下程式碼片段以 IHttpModule 實作 Filter 功能。

```
Public Class MyHttpModule Implements IHttpModule
…
Public Sub Init(ByVal context As System.Web.HttpApplication)
Implements System.Web.IHttpModule.Init
        ‘註冊以DoAcquireRequestState方法處理 Request
        AddHandler context.AcquireRequestState, AddressOf
Me.DoAcquireRequestState
End Sub
Private Sub DoAcquireRequestState(ByVal source As Object,
ByVal e As EventArgs)
        …
        Dim cookie1 As HttpCookie =
request.Cookies("cookie1")
        Dim cookie2 As HttpCookie =
request.Cookies("cookie2")
        '驗證 cookie 的一致性
        …
        '比對hash
```

```
            Dim salt As Byte() = _
Encoding.ASCII.GetBytes(ConfigurationManager.AppSettings(
"salt"))
            '使用預先設定的 salt 及 password 計算出 hash key
            Dim key As New
Rfc2898DeriveBytes(ConfigurationManager.AppSettings("pass
word"), salt)
             '將 key 代入 HMACSHA1演算法
            Dim myHash As New HMACSHA1(key.GetBytes(16))
myHash.ComputeHash(Encoding.ASCII.GetBytes(cookie1.Val
ue))
            If Not
Convert.ToBase64String(myHash.Hash).Equals(cookie2.Value
) Then
                Throw New ApplicationException("Cookie 一致性
驗證失敗")
            End If
            '使用Regular Expression尋找是否包含不合法字元
            Dim pattern As String = "<.+>"  '檢查參數中是否有
<XXX> 的輸入
            For Each field As String In request.Params.Keys
                For Each value As String In
request.Params.GetValues(field)
                    If Regex.IsMatch(value, pattern,
RegexOptions.IgnoreCase) Then
                        Throw New ApplicationException("輸入
包含非法字元: " + value)
                    End If
                Next
            Next
        End Sub
End Class
```

在 Filter 檢核可以阻擋大部份的不合法字元輸入，但是惡意使用者可能利用 URL Encoding 的方法來迴避；因此在呈現層我

們針對可以明確定義資料特徵的欄位以白名單方式進行檢核。以下程式碼片段顯示在 ASPX 頁面中使用.NET 內建的 RegularExpressionValidator 控制項驗證 Email 欄位的輸入。

```
<asp:TextBox ID="txtEmail" runat="server"
MaxLength="30"></asp:TextBox>
<asp:RegularExpressionValidator ID="valEmail"
runat="server"
    ControlToValidate="txtEmail"
    ErrorMessage="Email 格式不符"

ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\
w+)*">
</asp:RegularExpressionValidator>
```

履歷欄位的輸入驗證是我們最頭痛的，因為履歷內容可能出現各式各樣的字元；可能是英文、中文、數字、全型符號或半型符號，可能無法明確的定義出資料特徵，我們可能使用黑名單的方式檢核；但我們擔心黑名單方式可能無法涵蓋所有的可能性，因此履歷資料在輸出到頁面前應該被編碼；事實上不只履歷資料，所有資料在呈現前都應該被編碼。在此我們使用微軟推出的 Anti-Cross Site Scripting Library[8]。

```
‘以身份證字號取得使用者的履歷資料
Dim resumeStr As String =
ResumeManager.GetResumeContent(IdNumber)
'使用 Anti-Cross Site Scripting Library 在輸出前對資料內容作
編碼
ResumeContent.Text =
Microsoft.Security.Application.AntiXss.HtmlEncode(resumeSt
r)
```

商業邏輯層同樣也是一個適合實作資料驗證的選擇，由於不需要處理錯誤的資訊呈現，因此可以使用程式語言提供的函式來實作資料驗證。以下程式碼片段在商業邏輯層中使用 Date.TryParse 來驗證使用者輸入的生日欄位、以及使用 Date.Compare 來驗證求職者的年齡是否符合要求。

```
Dim dateBirthday As Date
If Not Date.TryParse(strBirthday, dateBirthday) Then
    Throw New ArgumentException("生日日期格式錯誤")
End If
If Date.Compare(New Date(Date.Now.Year - 30, 1, 1),
dateBirthday) < 1 Then
    Throw New ArgumentException("此職位求職者年齡應大於30歲")
End If
```

資料來到資料存取層時，如前面所述，我們已不建議再作資料驗證；能作的只有驗證必要欄位是否有值，還有使用參數式的資料庫查詢指令；如以下程式碼片段所示。

```
Public Function SaveResumeData(ByVal id As String, ByVal
birthday As Date, ByVal phone As String, ByVal email As
String, ByVal resumeContent As String) As Integer
        '檢查必填欄位是否為空值
        If String.IsNullOrEmpty(id) Then
            Throw New ArgumentNullException("ID 不可為空值")
        End If
        If String.IsNullOrEmpty(resumeContent) Then
            Throw New
ArgumentNullException("ResumeContent 不可為空值")
        End If
        Dim connection As New
SqlConnection(ConnectionString)
```

```
        ‘使用  parameterized SQL Statement
        Dim sql As String = "INSERT INTO RESUMES
(USER_ID, BIRTHDAY, PHONE,
            EMAIL, RESUMECONTENT) VALUES (@uid,
@birthday, @phone, @email,
            @resume)"
        Dim command As SqlCommand = New
SqlCommand(sql, connection)
        command.Parameters("@uid ").Value = id
        …
        Return command.ExecuteNonQuery()
End Function
```

結論

在以往程式都是以單機方式（Standalone）執行的時代，資料驗證的議題並未受到重視；因為透過網路傳輸的資料有限，使用者的數目、甚至身份都是可以被預期的。然而現在大多數的系統都被發佈到網路上，只要有網址就能夠使用，我們已經無法預期使用者的數量，更無法保證使用者是否是善意的；然而從今年 8 月的 Mass SQL Injection 事件中，我們可以發覺到資料驗證的重要性在許多系統開發過程中仍然被忽視，導致網站出現漏洞，讓惡意使用者輕易得逞。驗證輸入資料的動作雖然不易實作，卻是保護自己最有效的方法。”所有輸入都是惡意的”，當系統從界限外接收每筆資料時，我們都應該謹記這句話。

參考資料

[1] Writing Secure Code, Second Edition
http://www.microsoft.com/mspress/books/5957.aspx
[2] OWASP Top Ten Project
http://www.owasp.org/index.php/OWASP_Top_Ten_Project
[3] Regular Expression
http://msdn2.microsoft.com/zh-tw/library/hs600312(VS.80).aspx
[4] OWASP: Cross site scripting
http://www.owasp.org/index.php/Cross_Site_Scripting
[5] OWASP: Injection Flaw
http://www.owasp.org/index.php/Top_10_2007-A2
[6] “How to Break MD5 and Other Hash Functions” Xiaoyun Wang; Hongbo Yu
http://www.infosec.sdu.edu.cn/uploadfile/papers/How%20to%20Break%20MD5%20and%20Other%20Hash%20Functions.pdf
[7] C14N - Canonicalization
http://en.wikipedia.org/wiki/C14n
[8] Microsoft Anti-Cross Site Scripting Library
http://msdn.microsoft.com/en-us/library/aa973813.aspx