ftp client library 實作筆記

---

rfc 959 內容摘要

ftp client library 的實作是一個網路程式設計的良好課題。閱讀 rfc 文件與整理先輩於網路之中所遺留下來的隻字片語，一步一步建構理解 ftp client 端與 server 端的互動方式。

於 rfc 959 之中，雖然已經有許多部分不被現在所使用的 ftp client / server 所實作，但基本的概念仍然是一致的。 ftp 是一個基於 telnet protocol 的通訊協定。但也只用到極少特殊的 telnet function，如: **telnet synch** function [1]。此外，還有一個重要的資訊: ftp client 與 ftp server 的通訊需要二種不同的頻道，PI (control connection) 與 DTP (data connection) [2]。

Data Connection 有二種不同的連線模式: 被動模式(PASV)與積極模式(Active)。在一般情況之下 ftp client 送出連線請求時， ftp server 假定這樣的連線為一般模式(即為非 PASV 的模式)。這二種的差異在於，ftp server 是否能直接連上 ftp client 所給的 ip / port pair 接著開始互動。這並不是總是可行的，因為 ftp client 可能在 NAT 的網路架構之下。這種情況就需要使

[1]實作上即為使用 TCP/IP urg flag ，此 urg flag 對應於 telnet protocol 的描述即為 out-of-band data；於 Java Socket 對應的 method 為 public void sendUrgentData(int data);

[2]此二種不同的頻道即為rfc 959 中所描述的 PI (protocol interpreter) 與 DTP (data transfer process)，實作上即在必要的時刻提供這二種頻道的 socket 連線。 PI 即為傳送 ftp command 所用的頻道， DTP 為傳送資料的頻道，例如: 檔案或顯示目錄的結果。

用 PASV 模式，由 ftp server 告知 ftp client 連線所需要的資訊 ip / port pair 的提供者轉換為 ftp server [3]。一般模式下，假設 ftp server 所提供的連線 port 為 21，那 ftp server 將會提供 port 20 的 data connection port 向 ftp client 請求建立 data connection 。而此之前 ftp client 則要在先提出 ftp client 指定的 port [4]讓 ftp server 接上。

ftp client 實作環境與完成度

因為絕大部分使用自己電腦的時間都處於 NAT 網路架構之下，故 library 僅實作測試 passive 模式的 data connection 。對於 ftp server 所傳回來的 response 僅簡單地判斷最近一個 ftp client 所傳送的 command 是否被 ftp server 接受。 此外，對於有使用到 out-of-band 傳遞方式的 command 暫時未列入實作項目。 以 telnet protocol 的觀 Þ ，server 端是否提供 out-of-band 是一種選擇性實作，如果支援 out-of-band 傳送方式則可當成一種恩惠。有些情況下不允許 ‘server’ 端提供這樣的功能，例如: 對方的 ftp server 是由純 java 實作的，無使用 java native interface 或其他 library 是無法得知是否 ‘client’ 傳來了 urgent data 。因此，這部分尚未列入必要性實作的部分。然而 rfc 959

[3]PASV 模式所提供的資訊大致為: 227 Entering Passive Mode (210,59,94,134,220,153) 由 6 組數字組成。 最前面的4組即為 IP 位址，後 2 組為 port 。 因為是 16-bit TCP port address 的表示方式，要轉換為一般表示法需要再做簡易的換算 $220 * 256^1 + 153 * 256^0 = 56473$。

[4]使用 PORT 指令，寫法是 6 組數字使用 ‘,’ 相隔，如同使用 PASV 指令所收到的那一串 ip + port 的表示法。

撰寫時並沒有考量到有字碼的問題，故沒有多做規範，目前的 ftp client 大多於程式之中額外實作成了簡單的轉碼功能[5]。

程式架構概說

ftp client 的實作大致上為實作 specification 中所列出的 ftp command，並撰寫 protocol interpreter (PI)，去傳送 ftp command，在適當的時候開啓 data connection 接收或傳送資料。中間當然還有一些與 ftp server 商討使用參數的問題，例如: 是否以 passive mode 連線？連線的 port 為何？ftp server 資料的表現方式(type I, or type A)[6]。

ftp client 的實作由 ProtocolInterpreter class 提供 ftp client 的連線。並使用 IConnector 傳送 ftp command ，在需要的時候取得 IDataConnector 傳送或接收資料。 IConnector 介面規範了一個 control connection 至少應具備的功能，在取得 ftp server 回應的部分，考量到轉碼的方便性，決定把 get method 的回傳型態以最原始的 byte[] 方式保存。而傳送 ftp command 用的 put method 接受 2 種參數，一種為原始的 command string ，一種為透過 ICommand 介面實作的實體變數。前者多半為測試用途，後者才為真正讓 library user 使用。

[5]轉碼的需求是重要的，當使者用發現下載後的中文檔名成為亂碼時，也許就會放棄使用這套軟體。自 unicode 提出而廣泛使用後，大多數的系統都紛紛支援，在這萬碼奔騰的時代融和過去與現在是必需的。

[6]雖然於 rfc 959 中列出了許多種組合，但目前大多數的 ftp server 只實作了 type I (image type) 與 type A (ASCII With No Print Form)

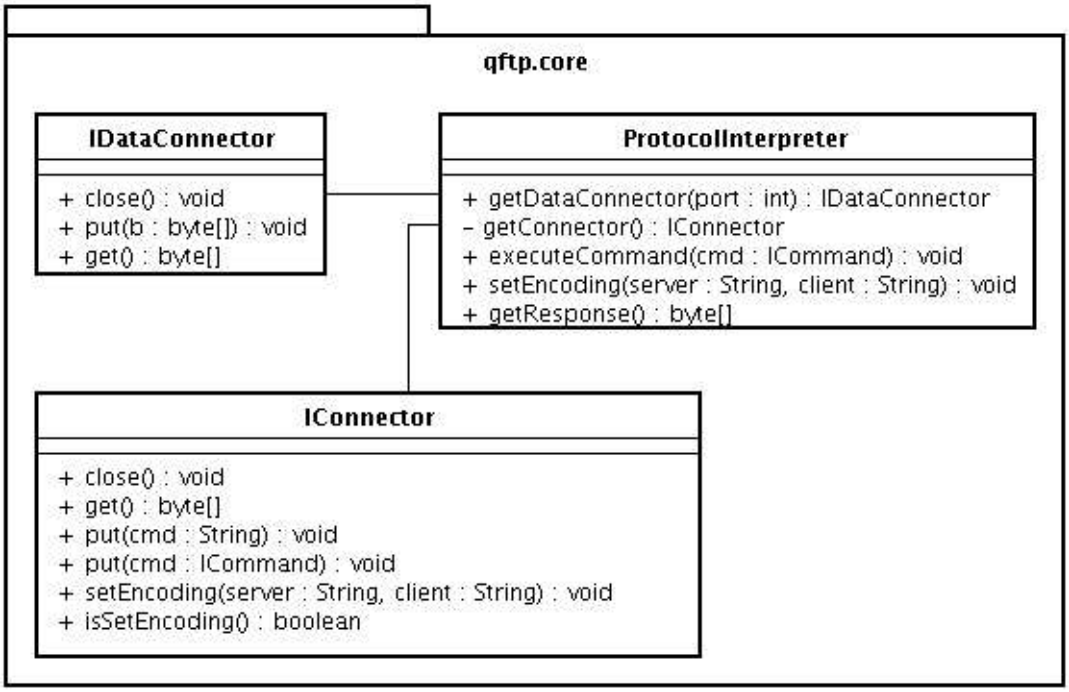

Figure 1: qftp.core package

經由 ProtocolInterpreter 連線後，將取得一私有的 IConnector ，令 IConnector 為私有是因為不希望 library user 直接使用 IConnector 避免不小心直接 close 掉 control connection 的連線。所有 ftp command 的執行皆透過 executeCommand method 間接呼叫 IConnector 。取得IDataConnector 的部分因為只實作了 passive mode 故只提供了有 ftp client 起始連線的方式 getDataConnector method 的參數需要解析 PASV 指令所回傳的解果來取得 port 。

而對於 ftp command 的組成由 ICommand 提供共同的介面，規範所有的命令必需覆寫 toString method 。依 rfc 959

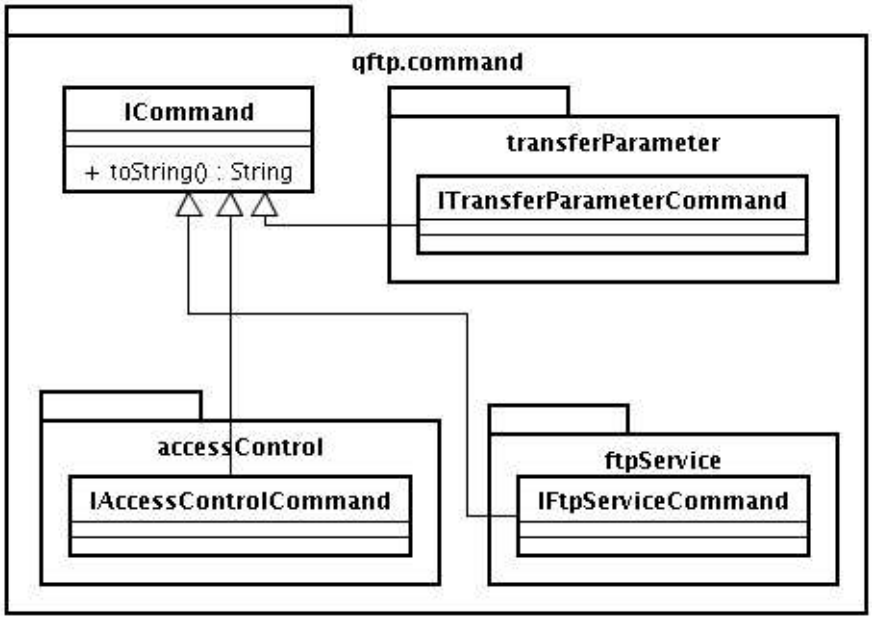

Figure 2: qftp.command package groups

中，命令的種類大致可分為與 ftp server 取得使用權限的 access control command ，協調傳輸參數的 transfer parameter command ，使用 ftp 上傳下載等服務的 ftp service command 。有些命令需要設定參數，我所採用的方式為 constructor initialization 方式，而非 setter method 設定參數值[7]。

[7]這裡存在改良的空間， ftp 指令並不會很多，也許可以改為 command pool 一開啓先產生常用的 command instance ，再提供 setter method 與 clean method 做設值與初始化的動作，可以減少一再產生物件的成本。

雜記

透過以 rfc 959 為藍本建構 ftp client 的過程，順便延伸閱讀了 telnet protocol 的想法。稍為明白了此二種 protocol 的異同。程式之中雖然不是非常符合物件導向的思考模式，但稍有其身形架構。本篇文件的撰寫是距離程式完成後的一個半月之後的事了。回頭看看自己寫的程式碼，大概只抓住了 ftp 實作的概念，但沒有寫出良好的 style ，與一些 interface 用的也許有些多餘。但我認為這是好事，今年八月才開始接處 design pattern 的我，是無法一步登天達到那完美的 style 。先讓一些概念出現在程式之中，隨著自己的成長讓他慢慢演化。