servlet form-based file upload 實作

rfc 1867[1] **內容摘要**

在初學 jsp/servlet 之時，對於檔案上傳套件只能抱著 “黑盒子” 的想法。但對於能看懂、能明白的程式碼與日俱增後，掩蓋不住的好心奇成為了實作檔案上傳功能的動機。 rfc 1867 主要規範了二件事: 為了上傳檔案新定義的 HTML DTD 與 user agent 應該如何傳送 “新式的表單”[2] 。

- FORM 的 **ENCTYPE** 屬性新增了檔案上傳用的值 “multipart/form-data” [3]
- 規範 user agent 如何傳送表單的 HTML string (listing 1)

[1] rfc 1867: Form–based File Upload in HTML

[2] 新式表單不單只使用上 INPUT 多了 FILE 屬性，更實際層面的改變是，提出第二種表單傳送的方式以因應巨大的串流資料。資料量不單只是傳傳文字那般，而是為了傳輸檔案而提出的新方法。

[3] 其實接在 multipart/ 之後的還有 mixed 等，但處理表單檔案上傳時大部分注意 form-data 的用法即可。 如果沒有填寫 ENCTYPE 時，規範上會填上預設值 ENCTYPE = "application/x-www-urlenocded"

Listing 1: Example for client html string

```
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="field1"

Joe Blow
--AaB03x
content-disposition: form-data; name="pics"; filename="file1.txt"
Content-Type: text/plain

... contents of file1.txt ...
--AaB03x--
```

實作筆記

在閱讀 rfc 1867 之後，至少明白了為什麼一般傳字串可以不加 ENCTYPE ，為什麼傳送檔案資料卻要自己加上 ENCTYPE 並明確指定為 multipart/form-data [4]。 當開始決定著手寫看看時，發生了一個問題(listing 2):

Listing 2: Q: How to get content size

```
public MultipartRequest(HttpServletRequest req){
        InputStream in = req.getInputStream();
        int contentSize = in.available();
}
```

我發現這樣做似乎不會回傳任何除了 0 以外的數值。但是真的去試著做 read 卻又有資料。百試不爽都是這個結果，開始查閱 SmartUpload 內是怎麼處理這一部分的。終於明白如何取得 content-length (listing 3)。

[4]加與不加其實不是很絕對的事，有的 user agent 容錯力強，自己補上或是猜到作者的“心意”。有的 user agent 保持原貌，忠於原味。這之中的好壞無法評斷，身為寫 code 的人，只能期許自己儘量依賴標準規範。

Listing 3: A: How to get content size

```
public MultipartRequest(HttpServletRequest req){
        InputStream in = req.getInputStream();
        int contentSize = req.getContentLength();
}
```

能取得了 user agent 所傳來的資料長度後，複雜的問題才真正開始。其實做 parse 的工作並不算太複雜，只是有些小細節要注意。首先透過觀察 raw html string (listing 4)來決定如何 parse ，觀察的結果如下:

- 由 boundary string 起始，並由 boundary string ‘-’ ‘-’ <CR><LF> [5]結束。
- 整份 ‘Content’ 內的容來看 pattern 為[6]:
  1. boundary string <CR><LF>
  2. Content-Disposition .* <CR><LF> (Content-Type .* <CR><LF>)?
  3. Content Body <CR><LF>
  4. **(1. 2. 3.)+** boundary string ‘-’ ‘-’ <CR><LF>

有了這樣的理解，就不難寫出 parser 來取得所需的內容。

[5]<CR><LF> 為 html string 中的分行記號，於 rfc 1867 中也有提及。

[6]條列項目中的 + * . 等符號請以正規表示式 (regular expression) 的符號去理解。

Listing 4: http string example from firefox

```
-----------------------------18091066257559025384284732222<CR><LF>
Content-Disposition: form-data; name="fn"; filename="2.cpp.orig"<CR><LF>
Content-Type: application/octet-stream<CR><LF>
<CR><LF>
#include <iostream>
#include <cstring>
#include <cstdlib>

using namespace std;

... 略 ...


<CR><LF>
-----------------------------18091066257559025384284732222<CR><LF>
Content-Disposition: form-data; name="name"<CR><LF>
<CR><LF>
d<CR><LF>
-----------------------------18091066257559025384284732222<CR><LF>
Content-Disposition: form-data; name="email"<CR><LF>
<CR><LF>
er<CR><LF>
-----------------------------18091066257559025384284732222--<CR><LF>
```

雜記

閱讀是取得資訊的方法之一，實作則是驗證與體會之道。在沒有親身實作 upload 之前，我大概會問為何用了 upload 套件就無法由 getParameter 取得變數(listing 5):

Listing 5: I cannot use getParameter method

```
String name = request.getParameter("name");
```

我想答案的線索也許在 rfc 1867 之中:

> When the user completes the form, and selects the SUBMIT element, the browser should send the form data and the content of the selected files. The encoding type application/x-www-form-urlencoded is inefficient for sending large quantities of binary data or text containing non-ASCII characters. Thus, a new media type, multipart/form-data, **is proposed as a way of efficiently sending the values associated with a filled-out form from client to server.**

理解上，二種傳輸方式同一次傳輸只能選擇一種。 developer 要在適當的地點自行接值。

然而，要以什麼樣的型式來應用這個 library 。一開始並沒有太大的計劃，但不知不覺寫成了裝飾者模式。也許剛好是本質上就很雷同: 我所做的努力，只是在幫 HttpRequest 做加工加料的動作，本質上 HttpRequest 還是 HttpRequest 並不會把他變成別種類別，只是在原有的類別上多加了新的功能。而這樣的手法，確實為裝飾者的意圖。