Lab 3

socket & regular expressions

一、 實驗目的

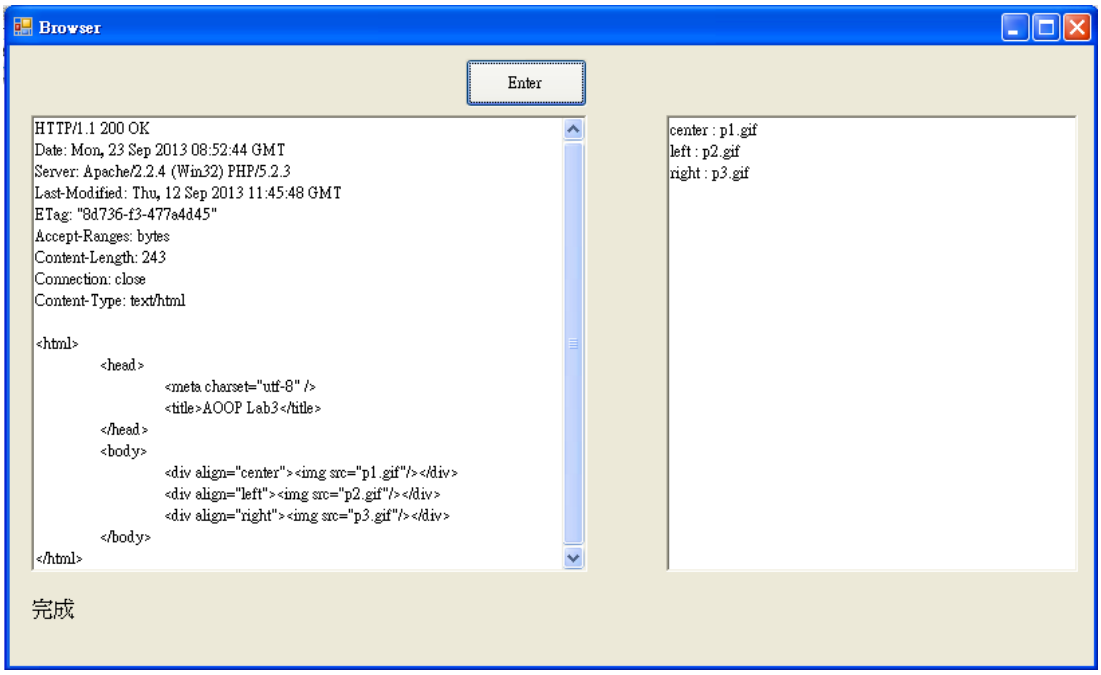
透過 visual C++ Windows Form 內建的 Socket、TCPListener 元件來進行 server和client 端的建立即連線，並透過 send 及 receive 函式，傳送 command 給server及接收來自server的訊息，最後透過 Regex 類別進行 html分析，達到最簡易版瀏覽器的效果。

二、 架構說明：

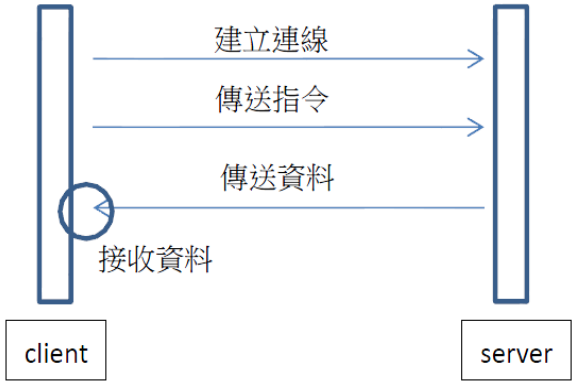
a. 使用工具元件：

Socket、Regular expressions、IO

b. 效果：



c. 運作方式：



三、 實際操作：

1. 此實驗必須使用 Socket、Text、RegularExpressions、IO 類別，請先加入命名空間

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;

using namespace System::Net::Sockets;
using namespace System::Text;
using namespace System::Text::RegularExpressions;
using namespace System::IO;
```

2. 建立連線

```
Socket ^mySocket;
mySocket = gcnew Socket(AddressFamily::InterNetwork, SocketType::Stream, ProtocolType::Tcp);
try
{
    mySocket->Connect("主機位置", port);
}
catch (Exception ^ee)
{
    label1->Text = "主機無法連接 !! " + ee->Message;
    return;
}
```

基本上網頁伺服器的預設 port number 都是 80

3. 傳送指令

```
String ^strTest = "GET "+"網頁名稱"+" HTTP/1.1\r\nHost: "+"主機位置"+"\r\nConnection: Close\r\n\r\n";
array<Byte>^bytesSent = Encoding::ASCII->GetBytes( strTest );
mySocket->Send(bytesSent);
```

4. 接收資料

```
String ^receivedata = nullptr;
array <Byte> ^myBufferBytes=gcnew array <Byte>(256);
int dataLength = 0;
do
{
    dataLength  = mySocket->Receive(myBufferBytes);
    receivedata += Encoding::ASCII->GetString(myBufferBytes, 0, dataLength);
    richTextBox1->Text = receivedata;
}while( dataLength > 0 );
mySocket->Close();
label1->Text="完成";
//write
File ^file;
file->WriteAllText("aoop_lab3.html",receivedata);
```

四、Regular expressions：

Regular Expression (以下簡稱 REGEX) 是以一組特定字元符號描述字串樣式規則的記述語法。簡單地說， REGEX 用於表達字元符號在字串中出現的規則。舉個例子說明，在 REGEX 中，字元 '^' 放在第一個位置表示字串開頭位置，當我寫下 ^A 的記述時，便表示必須是一個開頭為 A 的字串，如 Adam，才符合此一規則。這個表達規則通常稱為 pattern 。REGEX 規則是以 Perl 的規則為範本 (PHP 中稱為 PCRE 的內容) ，以下僅簡短說明幾個常見的字元意義。想了解更多用法者，請參看 Perl/PHP 的相關書籍。

1. ^
   寫在 pattern 第一個位置時，表示其後一符號必須出現在字串開頭的位置。寫在 pattern 中間位置時則為否定之意，表示字串中不可有 ^ 之後一符號的內容。
2. $
   寫在 pattern 最後一個位置時，表示其前一符號必須出現在字串尾端的位置。寫在 pattern 中時無特別意義。
3. *
   表示字串中有 0 到無數個其前一符號的內容。
4. +
   表示字串中有 1 到無數個其前一符號的內容。
5. ?
   表示字串中有 0 到 1 個其前一符號的內容。
6. { }
   表示前一符號在字串中的重覆次數。例如 “A{2}” 表示 'A' 重覆兩次 (即 'AA') ；”A{2,}” 表示字串含有 2 到無數多個 'A' ；”A{2,5}” 表示含有 2 到 5 個 'A' 。
7. .
   表示一個任意字元。
8. [ ]
   表示字串含有括號中任一字元的內容。可以 - 表示一組連續字元，例如 “[a-z]”, “[0-9]” 。注意， [] 僅代表一個字元，例如 “[abc]” 表示 'a' 或 'b' 或 'c' ，而不是 'abc' 。

9. ( )
   表示一個 sub pattern ，符合 sub pattern 的字串內容會被存放在匹配陣列中，並依序指派數字代表此 sub pattern 。可以此數字在 pattern 的其他地方引用內容，例如”The h([0-9]) means Title (\1)” 表示第 1 個 sub pattern 是 0 到 9 的任一字元，而 \1 表示匹配的內容。故 'The h1 means Title 1', 'The h2 means Title 2' 到 'The h9 means Title 9' 符合規則。
10. \
    表示轉義 (escaping) ，將其後的字元視為一般字元。例如要表示字串中含有 '/' 字元時，就必須寫作 “\/” 。
11. |
    「或」意，字串中含有 '|' 之前一符號或後一符號的內容。例如 ”image\.(jpg|png)” 表示 'image.jpg' 或 'image.png' 。通常會用 () 括住 '|' 的前後符號。
12. \d
    表示任何一個數字，意同 [0-9] 。
13. \D
    表示任何一個非數字，意同 [^0-9] 。
14. \w
    表示任何一個字元與數字以及 '_' ，意同 [a-zA-Z0-9_] 。
15. \W
    表示任何一個 \w 以外的字元。
16. \s
    表示任何一個空白符號，包括 \t, \v 等。
17. \S
    表示任何一個非空白符號。

要產生一個 Regex 個體的方式是向系統要求建立一個 Regex 個體，即 new Regex(pattern) ，引數 pattern 可以是一個字串也可以是另一個 Regex 個體。當要執行規則運算式作業時可以呼叫 Match 方法以擷取 Match 物件，即 Regex->Match(String)，這個方法所回傳的 Match 物件表示在字串或其中任何部分內的**第一個相符項目**。

Example:

pattern = “ABC”

Match ->Groups[0]->Value = “ABC”

pattern = “(ABC)”

Match ->Groups[0] ->Value = “ABC”

Match ->Groups[1] ->Value = “ABC”

pattern = “(A)B(C)”
Match ->Groups[0] ->Value = “ABC”
Match ->Groups[1] ->Value = “A”
Match ->Groups[2] ->Value = “C”

testingString = “<title> AOOP Lab3 </title>”
pattern =”<title>(.+)</title>”
Match ->Groups[0] ->Value = “<title> AOOP Lab3 </title>”
Match ->Groups[1] ->Value = “ AOOP Lab3 ”