第二十二章
網路程式設計

本章學習目標

- 認識網路
- 學習如何取得文件的內容資訊
- 學習如何建立 socket 連線
- 學習如何建立 TCP 伺服程式與客戶程式

22.1 網址與 InetAddress 類別的使用

IP 位址是以 4 個 8 bits 的數值，以 10 進位來表示，用來區分網路上的每一台電腦。

例如，銘傳大學放置 Web 主機的 IP 位址為

```
140.131.73.20
```

只要在瀏覽器內鍵入

```
http://140.131.73.20
```

即可連到銘傳大學的首頁。

- ✓ 習慣上會把電腦取一個簡單易記，且能代表此一電腦的名稱，這個名稱稱為 host name（主機名稱）。
- ✓ Host name 雖然好記，但電腦只認得 IP 位址，於是有 DNS（Domain name service）伺服器的產生。
- ✓ DNS 伺服器可以將 host name 轉成相對應的 IP 位址。

InetAddress 類別

Java 以 InetAddress 類別來處理有關 host name 與 IP 位址的取得。

下表列出了 InetAddress 類別常用的 method：

表 22.1.1 java.net.InetAddress 常用的 method

method	主要功能
static InetAddress[] getAllByName(String host)	給予電腦的 host name，取得該主機下所有提供服務的 IP 位址
static InetAddress getByName(String host)	給予電腦的 host name，取得該主機的 IP 位址
String getHostAddress()	取得電腦的 IP 位址
String getHostName()	取得電腦的 host name
static InetAddress getLocalHost()	取得本地端電腦的host name與 IP 位址

下面的程式碼是取得本機的名稱與 IP 位址之範例：

```
01  // app22_1, 取得本機的名稱與 IP 位址
02  import java.net.*;
03
04  public class app22_1
05  {
06     public static void main(String args[])
07     {
08        try
09        {
10           InetAddress adr=InetAddress.getLocalHost();
11           System.out.println(adr.getHostAddress());
12           System.out.println(adr.getHostName());
13           System.out.println(adr);
14        }
15        catch(UnknownHostException e) // 捕捉由 InetAddress()拋出的例外
16        {
17           System.out.println("無法取得 IP 位址");
18        }
19     }
20  }
```

```
/* app22_1 OUTPUT-----
169.254.180.217
buffalo
buffalo/169.254.180.217
----------------------*/
```

InetAddress 類別裡的 getByName() method 也允許由電腦 host name 取得相對應的 IP 位址，如下面的範例：

```
01  // app22_2, 取得主機的名稱與 IP 位址
02  import java.net.*;
03
04  class app22_2
05  {
06     public static void main(String args[])
07     {
08        try
09        {
10           InetAddress adr;      // 宣告 InetAddress 類別型態的變數 adr
11           adr=InetAddress.getByName("udn.com");    // 取得 IP 位址
12           System.out.println(adr);
13        }
14        catch(UnknownHostException e)
15        {
16           System.out.println("無法取得 IP 位址");
17        }
18     }
19  }
```

```
/* app22_2 OUTPUT-----
udn.com/210.244.31.152
----------------------*/
```

22.2 認識 URL

URL 是 universal resource locator 之意，用來表示網路上資源的位址。

下面的網址便是一個 URL 的範例：

```
http://udn.com:80/NEWS/main.html
```

Java 以 URL 類別來處理 URL 相關的資訊：

表 22.2.1 java.net.URL 常用的建構元與 method

建構元	主要功能
URL(String spec)	由字串 spec 建立 URL 物件
URL(String protocol, String host, String file)	以通訊協定、host name 與檔案路徑的字串建立 URL 物件
URL(URL context, String spec)	以一個絕對路徑的 URL 物件，以及相對路徑的檔案名稱建立 URL 物件

method	主要功能
Object getContent()	取得 URL 檔案的內容
String getFile()	取得 URL 的檔案名稱
String getHost()	取得 URL 的 host name
String getPath()	取得 URL 的路徑
int getPort()	取得 URL 的埠號
String getProtocol()	取得 URL 的通訊協定名稱
URLConnection openConnection()	建立一個 URL 連線，並傳回 URLConnection 物件

下面的範例說明了如何建立一個 URL 物件，及如何利用 URL 物件來取得該物件裡的資訊：

```
01  // app22_3, 使用 URL 類別
02  import java.net.*;
03
04  public class app22_3
05  {
06     public static void main(String args[])
07     {
08        try
09        {
10           URL u=new URL("http://udn.com/NEWS/main.html");
11
12           System.out.println("通訊協定名稱為 "+u.getProtocol());
13           System.out.println("host name 為 "+u.getHost());
14           System.out.println("埠號為 "+u.getPort());
15           System.out.println("檔名為 "+u.getFile());
16        }
17        catch(MalformedURLException e)
18        {
19           System.out.println("發生了 " +e+ "例外");
20        }
21     }
22  }
```

```
/* app22_3 OUTPUT-----
通訊協定名稱為 http
host name 為 udn.com
埠號為 -1
檔名為 /NEWS/main.html
----------------------*/
```

建立 URL 物件之後，即可利用 URL 物件呼叫 getContent() method，取得 URL 內的資源，如下面的範例：

```
01  // app22_4, 載入 URL 的檔案內容
02  import java.net.*;
03  import java.io.*;
04
05  public class app22_4
06  {
07     public static void main(String args[])
08     {
09        String str;
10
11        try
12        {
13           URL u=new URL("file:/c:\\java\\poem.txt");
14
15           Object obj=u.getContent();    // 取得 URL 的內容
16           InputStreamReader isr=new InputStreamReader((InputStream) obj);
17           BufferedReader br=new BufferedReader(isr);
18
19           while((str=br.readLine())!=null)
20              System.out.println(str);
```

```
21            br.close();
22         }
23         catch(IOException e)
24         {
25            System.out.println("發生了 "+e+"例外");
26         }
27      }
28   }
```

```
/* app22_4 OUTPUT----
床前明月光，
疑是地上霜。
舉頭望明月，
低頭思故鄉。
--------------------*/
```

使用 URLConnection 類別

如果要取得檔案的大小與類型等資訊，可利用 URLConnection 類別。

下表列出了兩個 URLConnection 類別所提供的 method：

表 22.2.2 java.net.URLConnection 常用 method

method	主要功能
int getContentLength()	取得資料所佔的位元數
int getContentType()	取得資料的型態

下面的程式碼說明了如何利用 getContentLength() 取得檔案的大小：

```
01  // app22_5, 使用 URLConnection 類別
02  import java.net.*;
03  import java.io.*;
04
05  class app22_5
06  {
07     public static void main(String args[])
08     {
09        try
10        {
11           URL u1=new URL("http://www.drmaster.com.tw");
12           URL u2=new URL("file:/c:\\java\\star.txt");
13           URL u3=new URL("file:/c:\\java\\pic0.jpg");
14
15           URLConnection uc1=u1.openConnection();
16           URLConnection uc2=u2.openConnection();
17           URLConnection uc3=u3.openConnection();
18
19           System.out.print("目錄頁的大小為 " + uc1.getContentLength());
20           System.out.println(", 類型為 " + uc1.getContentType());
21           System.out.print("star.txt的大小為 " + uc2.getContentLength());
22           System.out.println(", 類型為 " + uc2.getContentType());
```

```
23            System.out.print("pic0.jpg的大小為 " + uc3.getContentLength());
24            System.out.println(", 類型為 " + uc3.getContentType());
25         }
26         catch(IOException e)
27         {
28            System.out.println("發生了 "+e+"例外!");
29         }
30      }
31   }
```

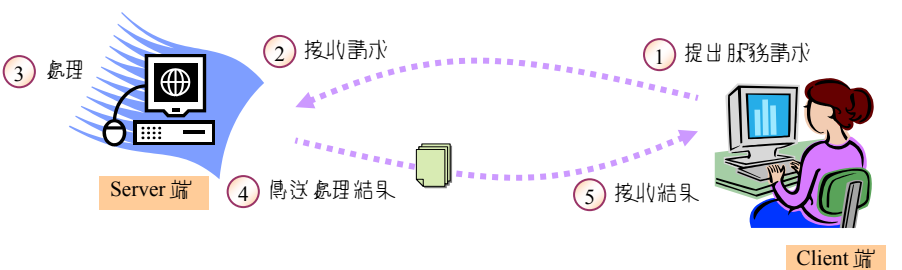
```
/* app22_5 OUTPUT----------------------

主網頁的大小為 1068,  類型為 text/html
star.txt 的大小為 62,  類型為 text/plain
pic0.jpg 的大小為 6280,  類型為 image/jpeg
----------------------------------------*/
```

22.3 建立主從架構程式 -- 使用 Socket 類別

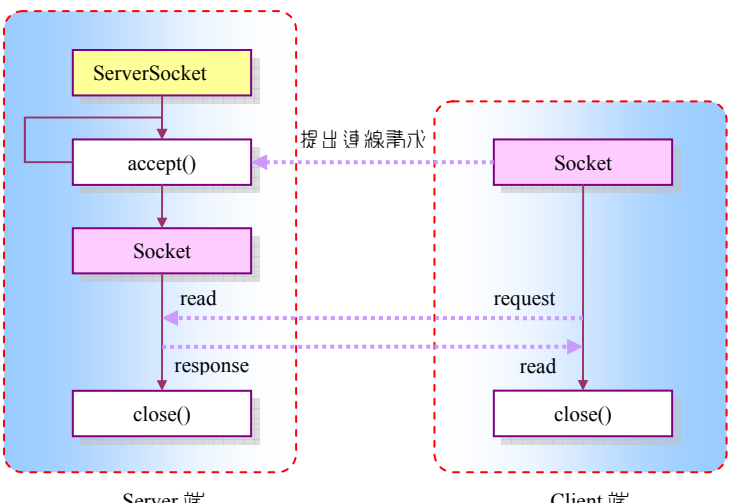
主從架構裡，不同的伺服程式會使用不同的埠號，同時在伺服器裡執行。

如有 client 端（客戶端）的請求送來時，則 server 端（伺服端）的伺服程式會對此一請求做出回應：



ServerSocket 與 Socket 類別，可將客戶端的電腦連上伺服器，並可從伺服器傳送資料給客戶端。

client-server 的運作的流程如下圖所示：



下面列出了 ServerSocket 與 Socket 類別所提供的建構元與 method：

表 22.3.1 java.net.ServerSocket 常用的建構元與 method

建構元	主要功能
ServerSocket(int port)	以埠號 port 建立 server socket 連線

method	主要功能
Socket accept()	監控客戶端的請求。當客戶端有請求時，便建立 socket 物件與客戶端連繫
void close()	關閉 socket

表 22.3.2 java.net.Socket 常用的建構元與 method

建構元	主要功能
Socket()	建立 socket 物件
Socket(InetAddress address, int port)	根據 IP 位址與埠號建立 socket 物件

method	主要功能
void close()	關閉 socket 連線
InetAddress getInetAddress()	取得 socket 所連線的 IP 位址
InetAddress getLocalAddress()	取得 socket 連線的本地端 IP 位址
InputStream getInputStream()	取得 socket 連線的輸入串流
OutputStream getOutputStream()	取得 socket 連線的輸出串流
int getLocalPort()	取得 socket 連線的本地端埠號
int getPort()	取得 socket 連線的遠端的埠號

建立 Server 端的伺服程式

本範例將建立一個 server 端的伺服端程式：

```
01  // app22_6, 建立 Server 端的伺服程式
02  import java.net.*;
03  import java.io.*;
04
05  public class app22_6
06  {
07     public static void main(String args[])
08     {
09        try
10        {
11           ServerSocket svs=new ServerSocket(2525);
12
13           System.out.println("等候客戶端的請求中 ...");
14           Socket s=svs.accept();   // 等候客戶端的請求
15           System.out.println("客戶端已 和本機連上線...");
16
17           OutputStream out=s.getOutputStream();   // 取得輸出串流
18           String str="Hello Java";
19           System.out.println("資料正 在傳送中 ...");
20           out.write(str.getBytes());// 將字串轉成 Byte 陣列,再寫入串流中
```

```
21            out.close();                    // 關閉輸出串流
22            s.close();                      // 關閉 socket
23            System.out.println("資料傳送完畢...");
24         }
25         catch(Exception e)
26         {
27            System.out.println("發生了 "+e+"例外!");
28         }
29      }
30   }
```

建立 Client 端的串流程式

接下來是一個客戶端的程式，撰寫如下：

```
01  // app22_7, 建立 Client 端的串流程式
02  import java.net.*;
03  import java.io.*;
04
05  public class app22_7
06  {
07     public static void main(String args[])
08     {
09        byte buff[]=new byte[1024];    // 建立 byte 型態的陣列
10        try
11        {
12           System.out.println("已在與串流器建立連線...");
13           Socket s=new Socket("127.0.0.1",2525);   // 建立 socket 物件
14           System.out.println("已經與串流器取得連線...");
15           InputStream in=s.getInputStream();       // 建立輸入串流
16           int n=in.read(buff);                     // 從串流讀入資料
17           System.out.print("從串流端收到: ");
18           System.out.println (new String(buff,0,n));// 印出讀入的內容
19           in.close();
20           s.close();
```

```
21          }
22          catch(Exception e)
23          {
24             System.out.println("發生了 "+e+"例外");
25          }
26       }
27    }
```

app22_6 與 app22_7 的執行結果如下所示：

```
/* app22_6 OUTPUT----

等候客戶端的請求中 ...
客戶端已和本機連上線...
資料正在傳送中 ...
資料傳送完畢...
-----------------------*/

/* app22_7 OUTPUT-------

正在與伺服器建立連線...
已經與伺服器取得連線...
從伺服端收到: Hello Java
-------------------------*/
```

-The End-