ξ3. System.Net

4. Yêu cầu và đáp ứng

Sau khi phân giải địa chỉ, client và server đã có thể liên lạc với nhau. Server tạo một socket và nghe ngóng sự kiện client kết nối đến. Client kết nối tới server và quá trình truyền nhận dữ liệu có thể bắt đầu. Bài học về socket sẽ đề cập chi tiết các hoạt động cần thiết. Ở đây ta chỉ hạn chế tới hai lớp WebRequest và WebResponse. Các lớp này tự chúng đã giải quyết các vấn đề liên quan tới socket. Cho nên, chúng rất dễ dùng.

Phương thức WebRequest.Create() được sử dụng để tạo WebRequest. Phương thức Create() nhận đầu vào là một đối tượng Uri, hay một chuỗi URI. Phương thức GetResponse() trả về đối tượng WebResponse và kết nối với server. Để đọc dữ liệu do server trả về, ta sử dụng luồng StreamReader để đọc từng dòng.

```
Uri uri = new Uri("http://www.wrox.com");
WebRequest request = WebRequest.Create(uri);
WebResponse response = request.GetResponse();
Stream stream = response.GetResponseStream();
StreamReader reader = new StreamReader(stream);
string line;
while ((line = reader.ReadLine()) != null)
{
  Console.WriteLine(line);
}
  response.Close();
  reader.Close();
```

Các lớp WebRequest và WebResponse

Các lớp cơ bản trong truy cập server Web gồm WebRequest và WebResponse.

WebRequest có các phương thức tĩnh và động cơ bản sau:

Các phương thức WebRequest tĩnh	Mô tả
Create() hay CreateDefault()	Lớp WebRequest không có hàm tạo nào thuộc dạng public. Thay vì thế, có thể tạo các bản sao bằng các phương thức tĩnh là Create() và CreateDefault(). Các phương thức này không thực sự tạo mới đối tượng kiểu WebRequest, mà là một đối tượng mới thuộc lớp WebRequest, chẳng hạn như HttpWebRequest hay FileWebRequest.
RegisterPrefix()	Bằng RegisterPrefix(), có thể đăng ký một lớp chuyên dùng để xử lý một giao thức cụ thể. Các đối tượng thuộc lớp này được tạo ra bằng phương thức WebRequest.Create(). Cơ chế này gọi là **pluggable protocol**.

Các phương thức WebRequest động gồm:

Các phương thức WebRequest động	Mô tả
GetRequestStream()	Trả về đối tượng luồng và ta dùng nó để gửi dữ liệu tới server.
BeginGetRequestStream() EndGetRequestStream()	Truy cập và chấm dứt truy cập vào luồng yêu cầu theo dạng bất đồng bộ.
GetResponse()	Trả về đối tượng WebResponse mà ta có thể dùng để đọc dữ liệu đã nhận được từ server gửi về.
BeginGetResponse() EndGetResponse()	Tương tự như luồng yêu cầu, đây là các phương thức nhận luồng đáp trả theo dạng bất đồng bộ.
Abort()	Nếu đã khởi động một phương thức bất đồng bộ có dạng BeginXX(), thì có thể chấm dứt hoạt động của nó bằng phương thức này.

Các thuộc tính của WebRequest:

Các thuộc tính của WebRequest	Mô tả
RequestUri	Thuộc tính chỉ đọc. Trả về URI đang phối hợp với WebRequest. URI này có thể được thiết lập bằng phương thức tĩnh Create().
Method	Dùng để đọc hoặc thiết lập phương thức cho một yêu cầu cụ thể. HttpWebRequest hỗ trợ các phương thức HTTP gồm GET, POST, HEAD, v.v...

Headers	Tùy thuộc vào giao thức đang dùng, nội dung header khác nhau có thể được truyền hay nhận từ server. Thông tin header của giao thức được lưu trong WebHeaderCollection và truy xuất được bằng thuộc tính Header.
ContentType ContentLength	Kiểu dữ liệu gửi tới server được xác định bằng thuộc tính ContentType. Ta có thể biểu diễn bất cứ kiểu khác nhau nào dưới dạng mảng các byte. Kiểu dữ liệu của nội dung thường xác định theo cách mà hình thức email MIME đang sử dụng, chẳng hạn image/jpeg, image/gif, text/html, hay là text/xml.
Credentials	Nếu server đòi hỏi người dùng phải chứng thực thì thuộc tính Credentials được sử dụng.
PreAuthenticate	Ngầm định, trình duyệt web thường cố truy xuất không kèm theo thông tin chứng thực. Nếu đòi phải có chứng thực, thì server trả về thông báo từ chối cấp phép truy cập. Trong yêu cầu tiếp theo từ client, trình duyệt sẽ phải đính kèm thông tin chứng thực. Có thể tránh phải áp dụng ngầm định không chứng thực của trình duyệt client bằng cách thiết lập sẵn thuộc tính PreAuthenticate là true.
Proxy	Thuộc tính này cho phép proxy đáp ứng yêu cầu của client thay cho server
ConnectionGroupName	Có thể định nghĩa một nhóm người dùng cùng sử dụng chung đối tượng WebRequest
Timeout	Xác định thời gian đợi đáp trả từ server, tính bằng millisecond. Ngầm định là 100.000. Nếu server không đáp trả WebException được thực hiện.

Lớp WebResponse được dùng để đọc dữ liệu của server. Một đối tượng của lớp này sẽ được trả về qua phương thức GetResponse(), như ta cũng thấy điều tương tự trong lớp WebRequest.

Các phương thức lớp WebResponse	Mô tả
GetResponseStream()	Trả về đối tượng luồng mà nó được dùng để đọc nội dung do server đáp trả.
Close()	Đóng đối tượng đáp trả khi nó không còn cần đến nữa

Các thuộc tính lớp WebResponse	Mô tả
ResponseUri	Cho phép đọc URI đang kết hợp với đối tượng đáp trả. Thuộc tính này giống URI trong đối tượng WebRequest, nhưng cũng có thể khác nếu server chuyển

	hướng yêu cầu tới một tài nguyên khác.
Headers	Trả về WebHeaderCollection mà nó bao hàm header chứa thông tin giao thức do server trả về.
ContentType hoặc ContentLength	Tương tự lớp WebRequest, các thông tin ở đây xác định kiểu loại dữ liệu được server trả về.

Để kết nối tới một server, cần phải có địa chỉ IP của nó. Do địa chỉ IP khó nhớ nên cần sử dụng dịch vụ tên miền (Domain Name Service). Bằng dịch vụ Whois hay tiện ích NSLookup ta có thể biết chút thông tin về server DNS ngầm định.

Lớp Dns dùng để phân giải tên các domain thành địa chỉ IP và ngược lại.

Phân giải tên thành địa chỉ IP

Để lấy địa chỉ IP từ tên của một host, ta có thể dùng phương thức Dns.Resolve(). Ta đã biết, một host có thể mang nhiều địa chỉ IP. Theo đó, Resolve() không chỉ trả về một dữ liệu kiểu IPAddress, mà nó còn có thể trả về đối tượng IPHostEntry. Đối tượng IPHostEntry lưu giữ cả một mảng địa chỉ, các bí danh và hostname tương ứng. Dưới đây ta thực hành việc phân giải địa chỉ IP của host có tên là www.microsoft.com bằng phương thức Dns.Resole(). Các địa chỉ IP của host được viết ra màn hình bằng cách truy cập thuộc tính AddressList của đối tượng IPAddress. Tiếp đó ta dùng vòng lặp, duyệt tất cả các bí danh có trong thuộc tính Aliases. Sau cùng ta viết tên thực của host và màn hình.

```
string hostname = "www.microsoft.com";
IPHostEntry entry = Dns.Resolve(hostname);
Console.WriteLine("IP Addresses for {0}: ", hostname);
foreach (IPAddress address in entry.AddressList)
   Console.WriteLine(address.ToString());
Console.WriteLine("\nAlias names:");
foreach (string aliasName in entry.Aliases)
   Console.WriteLine(aliasName);
Console.WriteLine("\nAnd the real hostname:");
Console.WriteLine(entry.HostName);
```

Lớp Dns còn có một số phương thức tĩnh khác cũng trả về đối tượng IPHostEntry. Chúng khác nhau ở cách truyền tên host. Để biết máy tính cục bộ mạng tên host là gì, ta có thể dụng phương thức Dns.GetHostName(). Phương thức này cũng trả về đối tượng IPHostEntry.

Các phương thức Dns	Mô tả
Resolve()	Chấp nhận một tên host hay địa chỉ IP dạng số thập phân phân cách bởi dấu chấm, phân giải thành địa chỉ IP
GetHostByName()	Chỉ chấp nhận tên host
GetHostByAddress()	Trả về đối tượng IPHostEntry khi được truyền địa chỉ IP hay chuỗi địa chỉ IP dạng dấu chấm phân cách thập phân. Nó còn chấp nhận cả đối tượng IPAddress.

Để có thể hiểu rõ hơn, người học cũng nên biết thêm về cách thức mà .NET phân giải tên miền thành địa chỉ IP và ngược lại. Trên thực tế có rất nhiều cách, liên quan tới dịch vụ thư mục, tới giao thức DHCP. Chưa hết, tên host còn dựa trên một giao thức có tên NetBIOS mà nó có thể dùng được trong mạng TCP/IP dưới dạng NBT-NetBIOS over TCP/IP.

Ứng dụng phân giải địa chỉ IP bất đồng bộ

Truy vấn một server DNS có thể tốn thời gian. Các phương thức đã nêu trên đều có dạng đồng bộ. Lớp Dns còn hỗ trợ phương thức truy vấn bất đồng bộ. Resolve() và GetHostByName() cũng có các phiên bản hỗ trợ bất đồng bộ. Sau đây ta sử dụng GetHostByName(), nhưng Resolve() cũng tương tự.

Các phiên bản bất đồng bộ của phương thức GetHostByName() gồm có BeginGetHostByName() và EndGetHostByName(). Phương thức BeginGetHostByName() bắt đầu thực hiện truy vấn tên nhưng không đợi đến thành công, cũng không trả về báo lỗi timeout. Ngoài việc truyền tên host, phương thức này chấp nhận cả một delegate AsyncCallback để thông báo về cho chương trình gọi ngay khi tên host được phân giải thành công hoặc lỗi timeout xảy ra. Ở đây ta dùng DnsLookupCompleted(), phương thức này được định nghĩa sẵn trong delegate AsyncCallback.

```
using System;
using System.Net;
class AsyncDnsDemo
{
   private static string hostname = "www.wrox.com";
   static void Main(string[] args)
   {
     if (args.Length != 0) hostname = args[0];
     Dns.BeginGetHostByName(hostname, new AsyncCallback(DnsLookupCompleted),null);
```

```
    Console.WriteLine("Waiting for the results...");
    Console.ReadLine();
  }
```

Ngay khi quá trình tra cứu DNS hoàn thành, phương thức DnsLookupCompleted() được triệu gọi và ta có kết quả truy vấn dựa trên lời gọi Dns.EndGetHostByName(). Sau đó ta truy xuất tất cả các địa chỉ IP, các bí danh và tên thực của host theo dạng bất đồng bộ sau đây:

```
  private static void DnsLookupCompleted(IAsyncResult ar)
  {
     IPHostEntry entry = Dns.EndGetHostByName(ar);
     Console.WriteLine("IP Addresses for {0}: ", hostname);
     foreach (IPAddress address in entry.AddressList)
        Console.WriteLine(address.ToString());
     Console.WriteLine("\nAlias names:");
     foreach (string aliasName in entry.Aliases)
        Console.WriteLine(aliasName);
     Console.WriteLine("\nAnd the real hostname:");
     Console.WriteLine(entry.HostName);
  }
}
```

Một cách khác để truyền delegate cho phương thức BeginGetHostByName(). Ta tham chiếu tới giao tiếp IasyncResult. Giao tiếp này trả về kết quả tra cứu DNS bằng cách sử dụng thuộc tính IsCompleted. Ngay khi phép tra cứu hoàn thành, như trên, ta cũng gọi phương thức đọc các địa chỉ IP và tên host: DnsLookupCompleted().

```
  static void Main(string[] args)
  {
     if (args.Length != 0) hostname = args[0];
     IAsyncResult ar = Dns.BeginGetHostByName(hostname, null, null);
     while (!ar.IsCompleted)
     {
        Console.WriteLine("Can do something else...");
        System.Threading.Thread.Sleep(100);
     }
        DnsLookupCompleted(ar);
```