ξ3. System.Net

Trong không gian System.Net, ta có các lớp lập trình mạng tra cứu địa chỉ IP, chứng thực mạng, cấp quyền và các lớp dùng cho truyền nhận dữ liệu. System.Net bao gồm các lớp con sau đây:

- **Nhóm tra cứu tên**: để tra cứu địa chỉ IP ứng với tên một host, hoặc để lấy tên host của một địa chỉ IP, ta có thể dùng lớp Dns.
- **Nhóm địa chỉ IP**: Các địa chỉ IP được xử lý qua lớp IPAddress. Một host có thể có nhiều địa chỉ IP và các bí danh. Các thông tin này được chứa trong lớp IPHostEntry. Lớp Dns sẽ trả về một đối tượng mang kiểu IPHostEntry khi chúng ta thi hành một phép tra cứu.
- **Nhóm chứng thực và cấp quyền**: lớp AuthenticationManager có chứa các phương thức nhằm chứng thực người sử dụng phía client đối với server. Lớp này sử dụng các module thi hành giao tiếp IAuthenticationModule. Các module làm nhiệm vụ chứng thực lấy thông tin chứng chỉ người dùng thông qua giao tiếp Icredentials và trả về cho phương thức gọi một đối tượng có kiểu Authorization. Ứng dụng client truyền thông tin người dùng cho server qua lớp NetworkCredential. Nhằm tăng tốc ứng dụng, lớp CredentialCache có thể được sử dụng để lưu trữ đệm các thông tin này.
- **Nhóm Yêu cầu và Đáp ứng**: các lớp cơ bản dành cho việc gửi các thông tin yêu cầu tới server (request) và nhận đáp ứng từ server (response) gồm có WebRequest và WebResponse. Trong không gian System.Net, có một số cách thức thi hành cụ thể để thực hiện các giao tiếp HTTP và truy cập tập tin, gồm: HttpWebRequest, HttpWebResponse, FileWebRequest, và FileWebResponse.
- **Nhóm quản lý kết nố**i: các lớp ServicePoint và ServicePointManager đóng vai trò quan trọng đối với các kết nối HTTP. Ta có thể tăng thông lượng của ứng dụng bằng cách tăng thêm kết nối. Ngầm định, số lượng kết nối tới cùng một dịch vụ server chỉ là hai. Nhóm quản lý kết nối cung cấp các phương thức để thay đổi số lượng kết nối.
- **Nhóm cookie**: cookie là một tập hợp dữ liệu lưu trữ ở phía client, nhưng được server sử dụng. Khi ứng dụng client yêu cầu thông tin tại server, trình duyệt gửi cookie tương ứng

cho server để chứng thực. Các lớp Cookie, CookieCollection và CookieContainer thực hiện các yêu cầu lập trình cần thiết.

- **Nhóm Proxy Server**: Các server proxy được dùng vì nhiều mục đích. Cơ bản nhất là sự hỗ trợ của nó trong môi trường mạng với phạm vi lớn để giảm thiểu thời gian trễ giữa client và server. Lớp WebProxy dùng để xác định proxy server. Lớp GlobalProxySelection để xác định proxy server ngầm định cho tất cả các yêu cầu của client khi chúng không chỉ định proxy server cụ thể.
- **Nhóm Socket**: mềm dẻo hơn, tận dụng được nhiều đặc điểm lập trình mạng hơn các lớp liên quan đến web. Lớp socket chủ yếu và quan trọng nhất có trong không gian System.Net.Sockets. Sử dụng các socket, ta có thể thi hành kết nối có hướng và vô hướng bằng cách sử dụng các kết nối broadcast và multicast. Socket còn rất mềm dẻo trong việc sử dụng các giao thức mạng như GGP, ICMP, IGMP, IPX và SPX.

1. URI

URI, viết tắt của Uniform Resource Identifiers**,** tạm dịch là *định danh tài nguyên*. Định danh tài nguyên trong mạng Internet là một trong rất nhiều vấn đề mà một hệ phân tán phải giải quyết. Ta sử dụng URI không chỉ trong việc truy cập tới các trang web, mà còn dùng trong việc truy cập tới các server FTP, trong các dịch vụ web và cả trong việc truy cập các tập tin cục bộ.

URI được định nghĩa trong RFC 2396. Cùng với URI có hai thuật ngữ khác gồm URL và URN.
URL (Uniform Resource Locator) trong các tài liệu chuẩn hiện nay không còn được dùng nữa. Người ta dùng URI thay cho URL với ý nghĩa rộng hơn. URN (Uniform Resource Name) là một URI đã được chuẩn hóa, nó được dùng khi nói tới một tài nguyên độc lập trong mạng chứa nó.

Trước hết, ta phân tích các thành phần của một URI. Cấu trúc hay gặp nhất của một URI là:

```
http://www.wrox.com  hay mailto:christian@nagel.net
```

Cấu trúc đầy đủ nhất của URI có dạng:

```
http://www.globalknowledge.net:80/training/generic.asp?pageid=1078&country=DACH
```

- Phần đầu của URI là dạng dịch vụ, tức scheme. Nó xác định không gian làm việc của URI đồng thời giới hạn cú pháp phải tuân thủ. Dấu :// để phân cách dạng dịch vụ với phần còn lại của URI.
- Sau dấu :// là địa chỉ IP hay tên của server. Ở đây là www.globalknowledge.net.
- Sau tên hay địa chỉ server là số hiệu cổng kết nối tới server. Nếu ta không chỉ rõ số hiệu cổng dịch vụ thì giao thức dịch vụ sẽ gán cổng ngầm định, chẳng hạn giao thức HTTP ngầm định dùng cổng 80.
- Sau địa chỉ cổng là đường dẫn tới tài nguyên mong muốn. Ví dụ /training/generic.asp.
- Dấu ? phân cách phần đầu URI với phần truy vấn (querry). Trong ví dụ, truy vấn được xác định là pageid=1078&country=DACH.
- Một tài nguyên mà truy vấn muốn truy cập tới có thể được phân chia thành các mảnh với các tên gọi riêng, hay bookmark trong các trang HTML. Ký tự # dùng để phân mảnh. Theo đó, nếu một URL có dạng http://www.microsoft.com/net/basics/glossary.asp#.NETFramework, thì mảnh mà truy vấn muốn truy cập đến là #.NETFramework.

Lưu ý là, nếu dấu # có mặt trong phần query thì đó không phải là để phân mảnh tài nguyên, mà nó là một bộ phận của chuỗi truy vấn. Trong một URL, ta có thể chứa query, hay một mảnh, nhưng không phải cả hai. Một số dấu có thể sử dụng trong URI, mỗi ký tự đều có chức năng riêng. Chúng bao gồm: ; / ? : @ & = + $ và ,

Lớp Uri

Lớp Uri thuộc lớp cha System. Nó có các thuộc tính và phương thức để truyền tham số, so sánh và tổ hợp các URI.

Ta có thể tạo đối tượng Uri bằng cách truyền chuỗi URI cho hàm tạo:

```
Uri uri = new Uri("http://msdn.microsoft.com/code/default.asp");
```

Nếu ta đã có một đối tượng Uri cơ bản thì có thể tạo mới URI bằng cách tổ hợp URI cơ bản với một URI khác:

```
Uri baseUri = new Uri("http://msdn.microsoft.com");
Uri newUri = new Uri(baseUri, "code/default.asp");
```

Nếu URI cơ bản đã có chứa sẵn đường dẫn thì nó có thể bỏ qua khi tạo mới đối tượng URI liên quan. Khi tạo URI liên quan, chỉ cần hình thức, số hiệu cổng và tên server tham gia dưới dạng các tham số tạo mới.

Kiểm tra tên host và hình thức dịch vụ

Lớp Uri có các phương thức tĩnh để kiểm tra tên của một host và dịch vụ cung cấp. Phương thức Uri.CheckschemeName() trả về giá trị true nếu có tồn tại dịch vụ, và Uri.CheckHostName() vừa kiểm tra tên host đồng thời trả về kiểu của host theo kiểu liệt kê UriHostNameType. Chúng có thể là một trong các giá trị sau:

Giá trị UriHostNameType	Mô tả
Basic	Tồn tại tên host, nhưng không xác định được kiểu.
Dns	Kiểu này thường được trả về
IPv4	Nếu chuỗi trả về dưới dạng địa chỉ bốn phần, phân cách bởi các dấu chấm, đó là kiểu IPv4.
IPv6	Nếu chuỗi trả về dưới dạng 1080:0:0:0:8:800:200C:417A, đó là kiểu IPv6.
Unknown	Giá trị này được trả về nếu địa chỉ host có chứa các ký tự không hợp lệ

Có thể dùng Uri.CheckHostName() để kiểm tra tính hợp lệ của chuỗi tên do người dùng nhập vào, nhưng nó không cho biết tên host có tồn tại trên thực tế hay không và có thể đến được hay không. Sự trợ giúp của lớp Dns rõ ràng hơn trong trường hợp này khi ta thực hiện chuyển đổi từ tên một host sang địa chỉ IP.

Các tính chất của lớp Uri

Lớp Uri có nhiều thuộc tính dạng chỉ đọc. Ta sẽ dùng địa chỉ sau để mô tả các thuộc tính này trong bảng dưới đây.

```
http://www.globalknowledge.net:80/training/generic.asp?pageid=1078&country=DACH
```

Thuộc tính	Mô tả
AbsoluteUri	Địa chỉ URI tuyệt đối mô tả đầy đủ một URL. Nếu có chỉ rõ số hiệu cổng trùng với cổng chuẩn quy định trong giao thức thì hàm tạo Uri tự gỡ bỏ. Khi đó, thuộc tính trả về giá trị như sau: http://www.globalknowledge.net/training/generic.asp?pageid=1078country=DACH Nếu tên tập tin được truyền cho hàm tạo của lớp Uri, thì AbsoluteUri tự động chèn thêm trước tên tập tin một tiền tố file://.
Scheme	Là hình thức dịch vụ mà ta thấy nó ngay đầu dòng. Ở đây là http.

Host	Thuộc tính này thể hiện tên host của URI. www.globalknowledge.net
Authority	Cũng giống như thuộc tính Host nếu số hiệu cổng trùng với quy định của giao thức. Nếu dùng một số hiệu cổng khác thì thuộc tính này chỉ ra cả tên host và số hiệu cổng.
HostNameType	Kiểu tên host. Kết quả giống như sử dụng với kiểu liệt kê UriHostNameType. Trường hợp này có kết quả là UriHostNameType.Dns
Port	Sử dụng thuộc tính này để đọc số hiệu cổng. Ở đây là 80.
AbsolutePath	Đường dẫn đến tài nguyên, tính từ sau số hiệu cổng cho tới chuỗi truy vấn. Ở ví dụ này là `/training/generic.asp`
LocalPath	Đường dẫn cục bộ. Chính là `/training/generic.asp`. Như ta thấy trong một yêu cầu HTTP, không có sự khác biệt giữa AbsolutePath và LocalPath. Nhưng sự khác biệt sẽ thể hiện nếu tài nguyên được tham chiếu qua một mạng. Với một tài nguyên chia xẻ qua mạng, thuộc tính LocalPath trả về có dạng `file:\\server\share\directory\file.txt`, thì AbsolutePath lại trả về tên server và các tên gọi dưới dạng bí danh, chia xẻ.
Query	Chuỗi truy vấn theo sau đường dẫn. Ví dụ này là `?pageid=1078&country=DACH`
PathAndQuery	Thuộc tính này trả về tổ hợp cả đường dẫn và chuỗi truy vấn.

2. Địa chỉ IP

Trong mạng TCP/IP, mỗi máy tính cần phân biệt với máy khác dựa trên định danh duy nhất. Định danh này là địa chỉ IP. Các địa chỉ IPv4 có cấu trúc 32 bit, chia làm 4 đoạn, phân cách nhau bằng dấu chấm. Về bản chất, đây là cách viết cho dễ đọc và cũng dễ để phân biệt giữa mạng và mạng con.

Lớp IPAddress thuộc không gian System.Net, có chứa nhiều lớp con và hỗ trợ chức năng chuyển đổi qua lại dưới dạng byte. Để tạo đối tượng IPAddress ta có thể dùng phương thức tĩnh Parse():

```
IPAddress address = IPAddress.Parse("204.148.170.161");
```

Lớp IPAddress lưu trữ địa chỉ IP dưới dạng một số nguyên. Số nguyên này có thể truy cập bằng thuộc tính Address. Phương thức ToString() có thể đọc số nguyên này và trả về chuỗi địa chỉ IP dưới dạng phân cách bằng dấu chấm.

Các địa chỉ định nghĩa trước

Lớp IPAddress có một số biến public dạng chỉ đọc mà nó trả về các địa chỉ IP đã định nghĩa trước.

- IPAddress.None trả về địa chỉ mà giao tiếp mạng cần dùng. Lớp Socket thường sử dụng địa chỉ này để báo hiệu rằng server đang ở tình trạng không nghe ngóng yêu cầu từ client.
- IPAddress.Loopback trả về địa chỉ phản hồi 127.0.0.1. Địa chỉ phản hồi (loopback) đã được định nghĩa sẵn, nó dành cho các hoạt động truy xuất tài nguyên cục bộ từ máy host.
- IPAddress.Broadcast trả về địa chỉ IP broadcast. Ta dùng địa chỉ IP broadcast để gửi thông điệp tới mọi máy trong mạng cục bộ.
- IPAddress.Any. Một máy tính có thể có nhiều card mạng, nên có thể có nhiều địa chỉ IP. IPAddress.Any được sử dụng bởi một socket để nghe ngóng từ bất kỳ giao tiếp mạng nào.

Thứ tự byte địa chỉ

Kỹ thuật mạng chính là kết nối các máy tính với nhau. Các máy tính mạng có thể có kiến trúc khác nhau, hệ điều hành khác nhau. Theo đó, hai thuật ngữ được định nghĩa gồm **little endian** và **big endian**.

Little-end: byte thấp được lưu ở địa chỉ nhớ thấp. Ngược lại, **Big-end** lưu byte thấp ở địa chỉ cao. Các CPU tương thích Intel dùng Little-endian, còn các CPU Motorola lại dùng Big-endian. Lớp IPAddress có hai phương thức tĩnh để chuyển đổi trình tự byte gồm IPAddress.NetworkToHostOrder() và IPAddress.HostToNetworkOrder().

Các địa chỉ IP và số hiệu cổng sử dụng trong các socket tuân thủ Big-Endian. Thực ra, lớp Socket đã giúp bạn đương đầu với vấn đề địa chỉ Big hay Little-End. Dữ liệu gửi qua mạng dưới dạng nào thì ta không cần quan tâm. Nếu ta quan tâm đến các ứng dụng cụ thể, sử dụng các kiến trúc máy tính khác nhau, nhất thiết phải quan tâm đến trình tự byte.