ξ2. LUỒNG TRONG C#

Bài này làm nền móng cho tất cả các ví dụ từ đây về sau. Nếu nắm bắt tốt các kiến thức xử lý xuất nhập (I/O) sẽ giúp hiểu nhanh nội dung các ví dụ và cũng giúp đúc rút kinh nghiệm lập trình.

I/O được áp dụng cho việc truyền dữ liệu qua mạng cũng như truy xuất đọc ghi các ổ đĩa. Tiết này giới thiệu hoạt động I/O dưới dạng các luồng, gồm:

- Những hiểu biết cơ bản về luồng C#.
- Các luồng khác nhau của .NET.

1. Giới thiệu các luồng

.NET phát triển một kiến trúc dựa trên luồng (stream). Các thiết bị I/O là bất cứ thứ gì, từ máy in cho tới các ổ đĩa cứng và cả các board mạng. Thiết bị khác nhau thường có chức năng khác nhau. Có nghĩa là, về bản chất, không phải tất cả các luồng đều hỗ trợ các phương thức giống nhau. Nhưng với .NET, lập trình viên có thể hiểu luồng như sau:

> Luồng là cách trình bày trừu tượng về một thiết bị tuần tự để lưu trữ và rút trích dữ liệu theo từng byte một. Theo đó, dù là thiết bị gì thì luồng truy xuất cũng chỉ cần dùng một tiến trình tương ứng, với mã lệnh tương tự nhau. Lập trình viên được giải phóng khỏi các cơ chế khác nhau của từng loại thiết bị cụ thể.

.NET framework cung cấp nhiều class khác nhau cho các luồng. Mỗi luồng này có chức năng riêng, thi hành các hành động khác nhau. Điểm duy nhất giống nhau giữa các luồng là: chuỗi byte tuần tự. Nói cách khác, để hiệu chỉnh luồng, ta sử dụng các lệnh nhị phân I/O cơ bản. Chẳng hạn TextReader và TextWriter là các lệnh thi hành I/O theo ký tự, còn BinaryReader và BinaryWriter thì thực hiện các hành động I/O nguyên thủy.

Các thuộc tính luồng, chẳng hạn như CanRead(), CanSeek() và CanWrite() biểu thị các khả năng nhất định của một luồng cụ thể, mỗi khi áp dụng nó cho một thiết bị cụ thể.

Trong lập trình mạng, luồng NetworkStream quan trọng nhất. Ngoài ra, một luồng quan trọng khác là FileStream. Ít khi nào các luồng này làm việc độc lập, chúng thường phối hợp trong ứng dụng.

Luồng có thể được dùng ở một trong hai chế độ: đồng bộ, hoặc bất đồng bộ.

Luồng đồng bộ: (ngầm định). Khi thi hành phương thức, nó dừng tất cả các công việc khác tận cho đến khi hoàn tất công việc, hoặc xảy ra một lỗi nào đó. Luồng đồng bộ phù hợp với kích thước truyền gửi nhỏ.

Luồng bất đồng bộ. Không cản trợ sự thi hành của các phương thức khác. Bất đồng bộ rất hiệu quả với các ứng dụng cần tiếp tục thực hiện các công việc khác trong khi đang phải xử lý một luồng có lượng dữ liệu lớn, hoặc là trong khi giao tiếp với một luồng thiết bị có tốc độ chậm.

2. Lớp luồng Stream cơ bản

Lớp Stream là một lớp cơ bản, có mặt trong không gian System.IO. Nó là lớp cha của một tập hợp các lớp luồng con khác. Nói cách khác, đa số các lớp luồng con có sự thừa kế của lớp cha Stream, từng lớp con có sự phát triển để tạo nên các hiệu ứng riêng. Hình T3.1 cho thấy toàn cảnh.

Các thành viên của luồng stream

Thuộc tính

Bảng sau liệt kê cácluồng con củaStream bao gồm 5 luồng con và ba thuộc tính chung:

Thuộc tính	Mô tả
CanRead	Được sử dụng nếu muốn kiểm tra luồng hiện hành có hỗ trợ đọc nội dung của nó hay không. Nhìn chung, thuộc tính này được sử dụng trước khi thực hiện các hành động đọc luồng. Giá trị trả về: true hoặc false. Nếu giá trị trả về là false thì còn kèm thêm một thông điệp NotSupportedException.
CanSeek	Mỗi luồng thường có con trỏ, chỏ vào một vị trí nào đó trong luồng. Thuộc tính CanSeek kiểm tra luồng hiện hành có hỗ trợ chế độ dò tìm vị trí con trỏ luồng hay không. Giá trị trả về: true hoặc false. Nếu giá trị trả về là false thì còn kèm thêm một thông điệp NotSupportedException.
CanWrite	Thuộc tính này thường được kiểm tra trước khi thi hành các hoạt động ghi vào luồng hiện hành. Giá trị trả về: true hoặc false. Nếu giá trị trả về là false thì còn kèm thêm một thông điệp NotSupportedException.

Ngoài ra, còn hai thuộc tính khác:

Thuộc tính	Mô tả

Length	Trả về một giá trị kiểu long, cho biết chiều dài của luồng, tính theo byte. Thuộc tính này thường được dùng để xác định chiều dài luồng, hoặc vị trí đầu/cuối của một mảng byte, hay một bộ đệm đang tạm thời lưu trữ luồng.
Position	Dùng để thiết lập hoặc tìm kiếm vị trí con trỏ của luồng. Để dùng được thuộc tính này, luồng phải hỗ trợ chế độ dò tìm (thuộc tính CanSeek phải trả về giá trị true)

Phương thức

Phương thức Seek(). Được dùng để thiết lập vị trí con trỏ trong luồng theo dạng truy cập ngẫu nhiên. Seek() cần cung cấp hai tham số. Thứ nhất, một giá trị long chỉ ra vị trí tính từ "mốc giới". Thứ hai, giá trị SeekOrigin chính là "mốc giới" mà tham số thứ nhất tham chiếu. SeekOrigin có ba giá trị gồm SeekOrigin.Begin, SeekOrigin.Current và SeekOrigin.End. (*Tham khảo bài ví dụ*).

Các phương thức đọc và ghi luồng được chia thành hai tập riêng dành cho truy xuất luồng theo dạng đồng bộ hay bất đồng bộ.

Phương thức đọc, ghi đồng bộ

Phương thức	Mô tả
Read() và ReadByte()	Được sử dụng nếu muốn đọc nội dung từ một luồng theo dạng đồng bộ. Phương thức Read() đọc một số byte cụ thể. Đọc hết số byte này, nó đẩy con trỏ đến vị trí byte đã đọc được sau cùng. ReadByte() chỉ đọc từng byte, đọc xong nó cũng đẩy con trỏ lên một vị trí. Chú ý, phương thức Read() trả về 0 nếu nó đến cuối luồng, trong khi ReadByte() sẽ trả về -1.
Write() và WriteByte()	Được sử dụng nếu muốn ghi nội dung vào một luồng theo dạng đồng bộ. Phương thức Write() ghi một chùm byte vào luồng và đẩy con trỏ đến vị trí byte đã ghi được sau cùng. WriteByte() chỉ ghi vào luồng từng byte, ghi xong nó cũng đẩy con trỏ lên một vị trí.

Phương thức đọc, ghi bất đồng bộ

Phương thức	Mô tả
BeginRead() và BeginWrite()	Được sử dụng nếu muốn đọc nội dung từ một luồng theo dạng bất đồng bộ. Cả hai phương thức đều sử dụng 5 tham số. Thứ nhất, mảng byte để đọc/ghi. Thứ hai, một số nguyên chỉ vị trí bắt đầu đọc/ghi. Thứ ba, một số nguyên chỉ ra số lượng byte để đọc/ghi. Thứ tư, là một hàm tùy chọn, delegate[1] AsyncCallback, nó được gọi khi hoạt

1 Delegate: thường được dịch là "ủy nhiệm hàm", nhưng chuyên môn vẫn gọi là delegate. Nó tương tự như con trỏ trong C++. Mục đích sử dụng nó là để phù hợp hóa giá trị truyền gửi giữa đối tượng và phương thức.

	động đọc/ghi hoàn thành. Thứ năm, là một tham số do đối tượng người dùng cung cấp nhằm để phân biệt hành động đọc khác nhau của các yêu cầu đọc khác nhau. Cả hai phương thức đều sử dụng giao tiếp IasyncResul để thể hiện tình trạng của hoạt động bất đồng bộ.
EndRead() và EndWrite()	Được sử dụng nếu muốn ghi nội dung vào một luồng theo dạng đồng bộ. Phương thức Write() ghi một chùm byte vào luồng và đẩy con trỏ đến vị trí byte đã ghi được sau cùng. WriteByte() chỉ ghi vào luồng từng byte, ghi xong nó cũng đẩy con trỏ lên một vị trí.

Các phương thức quản lý luồng

Phương thức	Mô tả
Flush()	Xóa tất cả nội dung bộ đệm sau khi đã chuyển toàn bộ nội dung tới đích của đối tượng luồng Stream.
Close()	Giải phóng tài nguyên, đóng các tập tin và socket liên quan tới luồng. Nếu nó được gọi thì nó tự động thi hành Flush() trước
SetLength	Thiết lập chiều dài cần thiết của luồng hiện hành