Stream trong C#, kiến trúc stream

Stream là một cơ chế hỗ trợ đọc ghi dữ liệu đặc biệt trong C#. Các hoạt động đọc ghi dữ liệu với file hoặc qua mạng (lập trình socket) đều phải sử dụng đến các phiên bản riêng của Stream. Do vậy, trước khi bắt đầu học làm việc với file, bạn cần biết rõ về stream. Do kiến trúc stream khá phức tạp, với nhiều loại class kế thừa và sử dụng lẫn nhau, không nắm bắt được cấu trúc của stream bạn sẽ rất dễ dàng lạc vào một "ma trận" các class chồng chéo.

Bài học này sẽ giới thiệu với bạn thông tin chi tiết về kiến trúc stream, giúp bạn có cái nhìn hệ thống về stream trong C#. Nó là nền tảng để bạn có thể học làm việc với file hoặc lập trình mạng với socket.

NỘI DUNG CỦA BÀI [Ẩn]

Khái niệm và kiến trúc stream trong C#

Trong .NET framework, *luồng dữ liệu* (stream) là một thành phần trung gian giữa ứng dụng và nguồn dữ liệu (file, network, v.v.) và có vai trò:

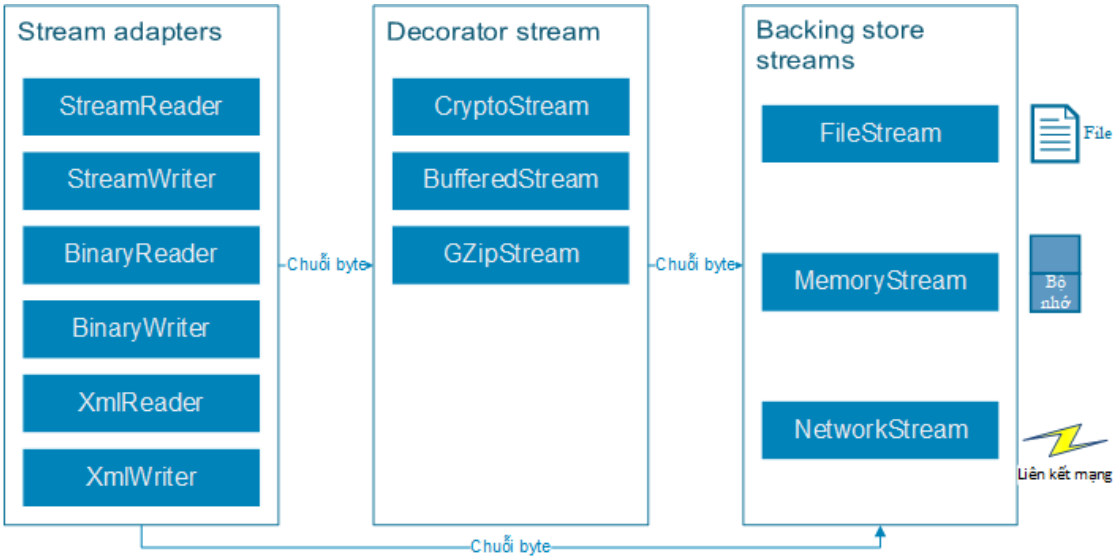
- hỗ trợ việc đọc/ghi dữ liệu với các loại nguồn khác nhau;
- cho phép sử dụng một khối lượng bộ nhớ nhỏ xác định để đọc dữ liệu với khối lượng lớn bất kỳ;
- giúp việc đọc và ghi dữ liệu ổn định, hiệu quả và đơn giản hơn.

Ví dụ, khi đọc dữ liệu từ một file lớn, nếu đọc toàn bộ dữ liệu cùng lúc, chương trình có thể treo vì không thể xử lý khối lượng dữ liệu quá lớn. Stream giúp đọc dữ liệu theo từng khối nhỏ hoặc từng byte riêng rẽ. Chương trình sau đó đọc dữ liệu từ stream.

Tương tự, khi cần ghi dữ liệu vào file, dữ liệu trước hết được đẩy vào stream. Sau đó, stream sẽ giúp ghi dữ liệu vào file. Tình huống tương tự cũng diễn ra khi đọc/ghi dữ liệu từ mạng (qua liên kết Tcp).

So sánh một cách hình tượng, stream giống như một đường ống nối chương trình với nguồn dữ liệu và cho phép một chuỗi byte (giống như dòng nước) chạy qua. Ngoài ra, stream cũng tạo ra một giao diện thống nhất để đơn giản hóa việc đọc/ghi dữ liệu.

Do số lượng lớp hỗ trợ làm việc với luồng dữ liệu trong .NET rất nhiều, chúng ta cần biết về kiến trúc của stream để tránh các nhầm lẫn khi sử dụng.



Kiến trúc stream trong .NET

Kiến trúc của stream trong .NET framework tương đối phức tạp và được chia làm ba nhóm: luồng làm việc với nguồn dữ liệu (backing store stream), luồng hỗ trợ (decorator stream), bộ điều hợp luồng (stream adapter).

Các thành phần của stream trong C#

Luồng làm việc với nguồn dữ liệu (backing store stream)

Nguồn dữ liệu (backing store) là nơi dữ liệu thực sự chứa dữ liệu. Luồng làm việc với các loại nguồn này chịu trách nhiệm đọc dữ liệu vào và/hoặc ghi ra theo từng chuỗi byte.

Có một số loại nguồn dữ liệu chính: file, bộ nhớ (memory), mạng (network), và vùng lưu trữ riêng (isolated storage). Do đó, C# tạo ra các loại stream khác nhau để làm việc với từng loại nguồn dữ liệu tương ứng, phân biệt là FileStream, MemoryStream, NetworkStream, IsolatedStorageStream.

Các loại stream này được gọi chung là *backing store stream* (luồng làm việc với nguồn dữ liệu).

> *Isolated Storage* là vùng lưu trữ riêng của các ứng dụng bị giới hạn quyền, ví dụ ứng dụng viết bằng Silverlight. Các ứng dụng này được cài đặt trên hệ thống nhưng có độ tin tưởng nằm trong vùng Internet, do đó bị giới hạn quyền truy xuất hệ thống file của windows.

Tất cả các lớp backing store stream của .NET đều kế thừa từ lớp abstract Stream (System.IO.Stream).

Trong bài học tiếp theo bạn sẽ học cách làm việc với FileStream. Nếu quan tâm đến NetworkStream, bạn có thể tìm đọc bài giảng lập trình mạng trong C#.

Luồng hỗ trợ (decorator stream)

Luồng hỗ trợ (decorator stream) không làm việc trực tiếp với nguồn dữ liệu mà làm việc với luồng backing store để cung cấp những chức năng hỗ trợ như mã hóa, nén hoặc tạo bộ đệm.

Nếu nhìn trong code thì object của luồng hỗ trợ chứa object của luồng backing store. Nói cách khác, luồng backing store luôn được khởi tạo trước, luồng decorator nhận luồng backing store làm tham số khi khởi tạo.

Tất cả hoạt động của luồng decorator tác động lên luồng backing store chứa trong nó. Các luồng hỗ trợ cũng có thể ghép nối với nhau, ví dụ, để vừa nén vừa mã hóa dữ liệu trong luồng backing store.

Trong tập bài giảng này chúng ta không sử dụng đến luồng hỗ trợ

Bộ tiếp hợp (stream adapter)

Các bộ tiếp hợp luồng (stream adapter) được sử dụng để chuyển đổi từ byte (do luồng backing store đọc) sang dữ liệu cấp cao giúp đơn giản hóa việc ghi đọc các dữ liệu này.

Luồng backing store và luồng decorator hoàn toàn hoạt động với byte hoặc mảng byte. Tuy nhiên, chương trình thường yêu cầu làm việc với dữ liệu cấp cao hơn như văn bản hoặc xml. Các bộ tiếp hợp cung cấp các phương thức chuyên dụng để làm việc với dữ liệu cấp cao theo từng loại định dạng cụ thể:

- lớp StreamWriter/StreamReader để làm việc với văn bản;
- lớp BinaryWriter/BinaryReader để làm việc với các kiểu dữ liệu cơ sở (int, bool);
- lớp XmlWriter/XmlReader làm việc với xml.

Nói tóm lại:

- luồng backing store cung cấp dữ liệu thô (các byte/khối byte);
- luồng decorator cung cấp các phương pháp biến đổi cho dữ liệu này;
- các adapter cung cấp các phương thức để chuyển đổi về dữ liệu cấp cao.

Các nhóm này phối hợp với nhau theo trình tự (khởi tạo object): backing store stream => decorator stream (có thể không sử dụng) => stream adapter (có thể không sử dụng nếu cần làm việc với các byte thô).

StreamWriter/StreamReader, BinaryWriter/BinaryReader được trình bày chi tiết trong bài học về FileStream hoặc NetworkStream (khóa học lập trình socket).

Lớp Stream

`Stream` là một lớp trừu tượng đưa ra quy định về một tập hợp các phương thức mà tất cả các lớp backing store stream phải thực thi.

Các phương thức này chia thành các nhóm: đọc, ghi, và định vị.

Vì lí do này, mặc dù chương trình đọc/ghi dữ liệu từ các nguồn khác nhau nhưng đều sử dụng các phương thức có mô tả giống nhau. Điều này giúp việc đọc ghi dữ liệu đơn giản hơn rất nhiều. Nếu sử dụng được một loại stream có thể dễ dàng sử dụng được các loại stream khác.

Tùy thuộc vào loại backing store, các luồng có thể hỗ trợ cả đọc/ghi/định vị, hoặc chỉ hỗ trợ một/một vài chức năng.

Ví dụ, FileStream hỗ trợ đủ ba chức năng, trong khi NetworkStream chỉ hỗ trợ đọc và ghi (không hỗ trợ định vị).

Ngoài ba nhóm chức năng trên, luồng hỗ trợ một số thao tác quản lý như đóng/mở luồng, đẩy dữ liệu (flush), định thời gian chờ.

Dưới đây là danh sách các thành viên chung của tất cả luồng (kế thừa từ lớp Stream):

Nhóm	**Thành viên**
Reading	public abstract bool CanRead { get; } public abstract int Read (byte[] buffer, int offset, int count) public virtual int ReadByte();
Writing	public abstract bool CanWrite { get; } public abstract void Write (byte[] buffer, int offset, int count); public virtual void WriteByte (byte value);
Seeking	public abstract bool CanSeek { get; } public abstract long Position { get; set; } public abstract void SetLength (long value); public abstract long Length { get; } public abstract long Seek (long offset, SeekOrigin origin);

Closing/ flushing	public virtual void Close();
	public void Dispose();
	public abstract void Flush();

Timeouts	public virtual bool CanTimeout { get; } public virtual int ReadTimeout { get; set; } public virtual int WriteTimeout { get; set; }
Nhóm khác	public static readonly Stream Null; // “Null” stream public static Stream Synchronized (Stream stream);

Kết luận

Qua bài học này bạn đã nắm được đầy đủ kiến thức về stream trong C#. Nó sẽ là nền tảng tốt để bạn có thể tự học lập trình với các loại lưu trữ khác nhau như file (sẽ học trong bài sau) hay lập trình mạng. Bài học này không hướng tới những kỹ thuật cụ thể, mà là lý thuyết hệ thống về stream.

+ Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
+ Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
+ Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
Cảm ơn bạn!