Chương 5

Lập trình đa luồng (Thread)

Mục lục chương

Giới thiệu

- Ở các chương trước chúng ta đã học cách sử dụng các lớp socket không đồng bộ để thực hiện các chức năng ở mức nền (background).
- Có nghĩa là chương trình sẽ tiếp tục chạy trong khi đợi các hàm socket thực hiện các chức năng của mình
- Chương này chúng ta sẽ học cách sử dụng thread để thực hiện các chức năng này.

Tạo thread trong chương trình

- C# cung cấp namespace System.Threading
- Trong đó có chứa các lớp cho việc tạo và điều khiển thread trong chương trình

Khái niệm thread

- Thread được định nghĩa như là một luồng đơn trong một chương trình
- Khi một chương trình thực hiện trên CPU, nó thực hiện qua các thread cho đến khi thread kết thúc
- Nếu một chương trình có nhiều thread thì sẽ có nhiều luồng được thực hiện đồng thời

183

Khái niệm thread

- Tất cả các thread được tạo ra share không gian nhớ với thread chính (main thread)

- Thông thường thread thứ cấp được tạo ra để thực hiện các tính toán trong khi main thread sẽ tiếp tục thực hiện các chức năng của chương trình

184

Lớp Thread

- Sử dụng lớp Thread để tạo ra một đối tượng thread và nhờ đó ta sẽ tạo ra một thread mới trong process hiện tại.
- Hàm tạo của lớp Thread:
  - Thread(ThreadStart start)
- Trong đó: ThreadStart là một ủy quyền và chỉ đến một phương thức sẽ thực hiện bên trong thread.

185

Ví dụ về tạo một thread mới

- Thread newThread = new Thread(new ThreadStart(newMethod));
- .
- .
- }
- void newMethod() { . . }

186

Các phương thức trong lớp Thread

Method	Description
Abort()	Terminates the thread
Equals()	Determines whether two Thread objects are the same
GetHashCode()	Gets a unique representation for the thread
GetType()	Gets the type of the current thread
Interrupt()	Interrupts a thread that is in the Wait thread state
Join()	Blocks the calling thread until the thread terminates
Resume()	Resumes a thread that has been suspended
Start()	Causes the operating system to change the thread state to Running
Suspend()	Suspends the execution of the thread
ToString()	Gets a string representation of the Thread object

Chạy một thread

- Sau khi một thread đã được tạo ra, ta phải gọi đến phương thức Start để bắt đầu chạy thread đó.
- Ví dụ về một chương trinh sử dụng Thread: ThreadSample.cs program

Sử dụng Thread trong Server

- Như vậy là chúng ta đã học cách tạo và sử dụng thread trong chương trình.
- Phần tiếp theo ta sẽ học cách sử dụng thread trong một server

189

Sử dụng Thread trong Server

- Một trong những thách thức chính khi ta xây dựng một server đó là làm sao để đáp ứng yêu cầu từ nhiều client
- Trong chương trước chúng ta đã sử dụng phương thức Select () để thực hiện điều này
- Tuy nhiên việc sử dụng Select () sẽ làm cho chương trình trở nên phức tạp và khó hiểu
- Việc sử dụng thread có thể khắc phục được những vấn đề đó

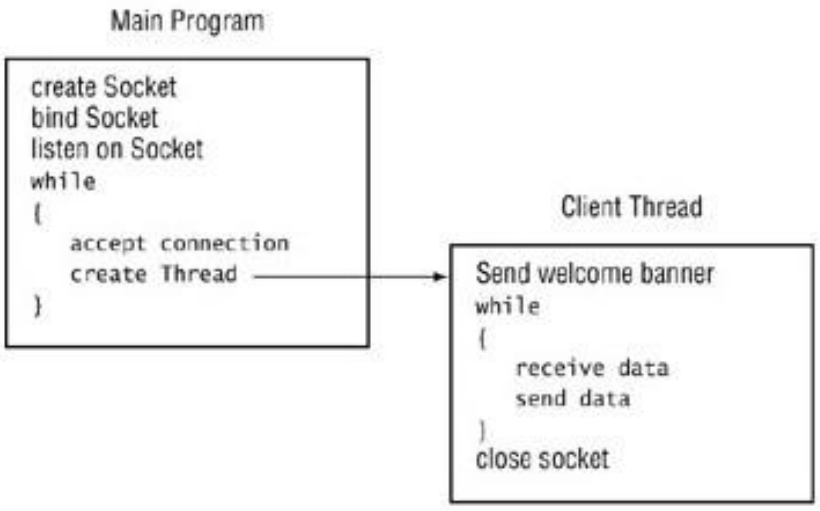
190

Tạo một Thread Server

- Điểm quan trong khi tạo một thread Server là mỗi khi một Client kết nối đến Server, chương trình chính trong Server sẽ tạo ra một thread để xử lý kết nối đó.

191

Tạo một Thread Server



192

Tạo một Thread Server

- Khi phương thức Accept() được gọi trong Server, nó sẽ tạo ra một đối tượng socket mới cho kết nối đó.
- Đối tượng này sau đó sẽ được truyền đến phương thức ủy thác của thread và sử dụng thread nới để trao đổi dữ liệu với Client.

193

Ví dụ về một thread Server

Ví dụ: ThreadedTcpSrvr.cs

- Ví dụ này cho phép chấp nhận một lượng không giới hạn các kết nối từ Client đến Server
- Điều này chưa hẳn đã tốt vì trong một số trường hợp nó có thể làm hết tài nguyên của máy tính

194

Testing the Server

- Sau khi chạy ví dụ trên, ta có thể thực hiện test server bằng cách chạy đồng thời một số Client và kết nối đến server
- Ta sẽ thấy rằng nhiều Client có thể kết nối đồng thời đến Server và có thể thực hiện truyền thông đồng thời đến Server

195

Sử dụng thread cho việc truyền và nhận dữ liệu

- Cùng với sự hỗ trợ cho việc kết nối nhiều Client, thread cũng có thể được sử dụng khi mà không có một mô hình giao thức rõ ràng nào được sử dụng.
- Một vấn đề là trong các ví dụ trước chúng ta đều biết rằng cả Client và Server biết chính xác là khi nào chúng sẽ nhận và truyền dữ liệu

196

Sử dụng thread cho việc truyền và nhận dữ liệu

- Tuy nhiên điều này là không đúng trong các ứng dụng thực tế
- Chẳng hạn như một chương trình chat thì bên nhận sẽ không biết chính xác là khi nào bên gửi sẽ gửi dữ liệu

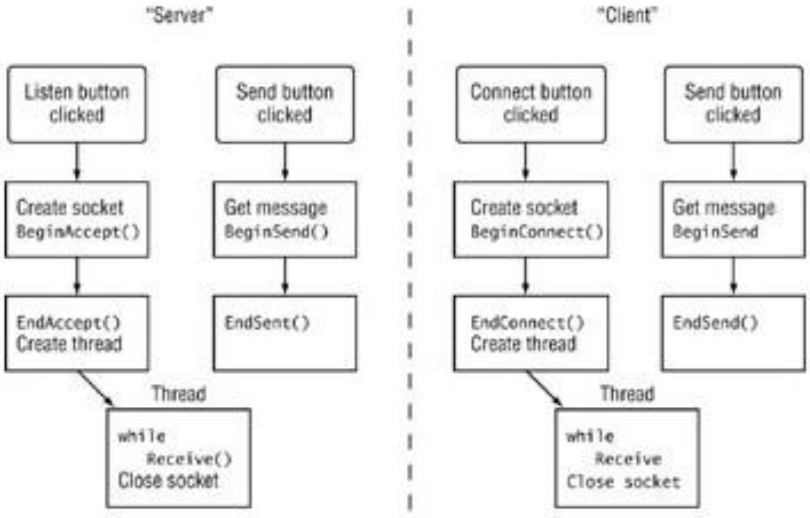
197

Sử dụng thread cho việc truyền và nhận dữ liệu

- Để giải quyết vấn đề này thì ta cần sử dụng Thread bằng cách tạo ra một thread thứ cấp chuyên để xử lý việc gửi và nhận dữ liệu

198

Sử dụng thread cho việc truyền và nhận dữ liệu



199

Ví dụ về một chương trình chat

- Vi dụ: TcpChat.cs
- Ví dụ này vừa có chức năng của một Client vừa có chức năng của một Server

200

Ví dụ - Chương trình chat

- Khi người sử dụng click vào nút lệnh Listen, một socket mới được tạo ra và lắng nghe các kết nối tới
- Khi có một kết nối tới, phương thức chuyển nó đến thread thứ cấp khác

  AcceptConn() tạo ra một socket mới và

Test chương trình chat

- Để test chương trình chat ta cần có copy ra 2 bản rồi chạy đồng thời
- Khi chạy chương trình này ta có thể chat
- Sử dụng chương trinh ListThreads ta có thể quan sát thấy các thread được tạo ra

Thread Pool

- Việc tạo ra nhiều thread mỗi khi có một kết nối đến sẽ làm chậm thời gian thực hiện của chương trình
- Để khắc phục tình trạng này ta dùng thread pool

203

Lớp ThreadPool

- Lớp ThreadPool được chứa trong System.Threading namespace
- Lớp ThreadPool gán một ủy thác tới các threads trong thread pool.
- Tất cả các phương thức trong lớp hàm tạo không bao giờ được sử dụng ThreadPool là phương thức tĩnh

204

Các phương thức trong lớp ThreadPool

Method	Description
BindHandle()	Binds an operating system handle to the thread pool
GetAvailableThreads()	Gets the number of worker threads available for use in the thread pool
GetMaxThreads()	Gets the maximum number of worker threads available in the thread pool
QueueUserWorkItem()	Queues a user delegate to the thread pool
RegisterWaitForSingleObject()	Registers a delegate waiting for a WaitHandle object
UnsafeQueueUserWorkItem()	Queues an unsafe user delegate to the thread pool but does not propagate the calling stack onto the worker thread
UnsafeRegisterWaitForSingleObject()	Registers an unsafe delegate waiting for a WaitHandle object

Đăng ký

- Để đăng ký một ủy quyền cho việc sử dụng các thread trong thread pool ta dùng cú pháp sau:
  - ThreadPool.QueueUserWorkItem(new
  - WaitCallback(Counter));

  - Biến *Counter* là ủy thác của phương thức chạy trong thread

Ví dụ về sử dụng ThreadPool

- Ví dụ: ThreadPoolSample.cs
- Test ví dụ về threadpool

207

Sử dụng ThreadPool trong Server

- Đối với các server mà có ít các client kết nối đến cùng một lúc (<25) thì ta có thể sử dụng ThreadPool để phục vụ cho các client thay vì sử dụng thread.
- Mỗi Client sẽ được gán một item và được xử lý bởi một thread trong threadpool

208

Ví dụ ThreadPool

-Ví dụ về Server sử dụng thread pool

- Quan sát các thread được tạo ra
- Test chương trình Server sử dụng threadpool

CHƯƠNG 6

GiAO TIẾP VỚI WEB SERVER