執行緒的停止

如果您想要停止一個執行緒的執行，當您查看API時，您會發現Thread的**stop()**方法已經被標示為 **"deprecated"**，使用這個方法來停止一個執行緒是不被建議的。

請見：**Why Are Thread.stop, Thread.suspend, Thread.resume and Runtime.runFinalizersOnExit Deprecated?**

如果您想要停止一個執行緒，您最好自行實作。

一個執行緒要進入Dead狀態，就是執行完run()方法，簡單的說，如果您想要停止一個執行緒的執行，就要提供一個方式讓執行緒可以執行完run()，而這也是您自行實作執行緒停止的基本概念。

例如，如果執行緒的run()方法中執行的是一個重複執行的迴圈，您可以提供一個flag來控制迴圈是否執行，藉此讓迴圈有可能終止、執行緒可以離開 run()方法以終止執行緒：

```
public class SomeThread implements Runnable {
  private boolean isContinue = true;

  public void terminate() {
    isContinue = false;
  }

  public void run() {
    while(isContinue) {
      // ... some statements
    }
  }
}
```

如果執行緒因為執行sleep()或是wait()而進入Not Runnable狀態，而您想要停止它，您可以使用interrupt()，而程式會丟出InterruptedException例外，因而使得執行緒 離開run()方法，例如：

- SomeThread.java

```
package onlyfun.caterpillar;

public class SomeThread implements Runnable {
    public void run() {
        System.out.println("sleep....going to not runnable");
```

```
        try {
            Thread.sleep(9999);
        }
        catch(InterruptedException e) {
            System.out.println("I am interrupted....");
        }
    }
}
```

- Main.java

```
package onlyfun.caterpillar;

public class Main {
    public static void main(String[] args) {
        Thread thread = new Thread(new SomeThread());
        thread.start();
        thread.interrupt();
    }
}
```

如果程式因為I/O而停滯，進入Not Runnable狀態，基本上您必須等待I/O完成才能離開Not Runnable，您無法使用interrupt()來使得執行緒離開run()方法，您要提供替代的方法，基本上的概念也是引發一個例外，而這個例外要 如何引發，要看您所使用的I/O而定，例如您使用readLine()在等待網路上的一個訊息，此時執行緒進入Not Runnable直到讀到一個訊息，您要讓它離開run()的方法就是使用close()關閉它的串流，這時會引發一個IOException例外而使得 執行緒離開run()方法，例如：

```
public class Client implements Runnable {
  private Socket skt;
  // .....

  public void terminate() {
    skt.close();
  }

  public void run() {
    // .....

    try {
```

```
        BufferedReader buf = new BufferedReader(
          new InputStreamReader(skt.getInputStream()));

        // 讀取客戶端訊息
        // 執行readLine()會進入Not runnable狀態
        // 直到讀到客戶端訊息
        while((userMessage = buf.readLine()) != null) {
          // ....
        }
      }
      catch(IOException e) {
        System.out.println("執行緒被終止.....");
      }
    }
  }
```

上面這個程式是個簡單的架構示範，實際的設計必須視您的程式功能與I/O類型而定。

除了stop()之外，**suspend()**、**resume()**方法也被標示為**"deprecated"**，這些方法如果您要達成相同的功能，您都必須自行實作，在將來新的Java版本中如果這些功能被實現，它也可能是新的介面，而不是使用現有的方法。

有關於執行緒的終止，還可以參考 **Two-phase Termination** 模式。