java创建线程的三种方式及其对比

一、Java中创建线程主要有三种方式：

1、继承Thread类创建线程类

（1）定义Thread类的子类，并重写该类的run方法，该run方法的方法体就代表了线程要完成的任务。因此把run()方法称为执行体。

（2）创建Thread子类的实例，即创建了线程对象。

（3）调用线程对象的start()方法来启动该线程。

```java
package com.thread;

public class FirstThreadTest extends Thread{
    int i = 0;
    //重写run方法，run方法的方法体就是现场执行体
    public void run()
    {
        for(;i<100;i++){
        System.out.println(getName()+"  "+i);
        }
    }
    public static void main(String[] args)
    {
        for(int i = 0;i< 100;i++)
        {
            System.out.println(Thread.currentThread().getName()+"  : "+i);
            if(i==20)
            {
                new FirstThreadTest().start();
                new FirstThreadTest().start();
            }
        }
    }
}
```

上述代码中Thread.currentThread()方法返回当前正在执行的线程对象。GetName()方法返回调用该方法的线程的名字。

2、通过Runnable接口创建线程类

（1）定义runnable接口的实现类，并重写该接口的run()方法，该run()方法的方法体同样是该线程的线程执行体。

（2）创建 Runnable实现类的实例，并以此实例作为Thread的target来创建Thread对象，该Thread对象才是真正的线程对象。

（3）调用线程对象的start()方法来启动该线程。

示例代码为：

```java
package com.thread;

public class RunnableThreadTest implements Runnable
{

    private int i;
    public void run()
    {
        for(i = 0;i <100;i++)
        {
            System.out.println(Thread.currentThread().getName()+" "+i);
        }
    }
    public static void main(String[] args)
    {
        for(int i = 0;i < 100;i++)
        {
            System.out.println(Thread.currentThread().getName()+" "+i);
            if(i==20)
            {
                RunnableThreadTest rtt = new RunnableThreadTest();
                new Thread(rtt,"新线程1").start();
                new Thread(rtt,"新线程2").start();
            }
        }
    }
}
```

线程的执行流程很简单，当执行代码start()时，就会执行对象中重写的void run();方法，该方法执行完成后，线程就消亡了。

3、通过Callable和Future创建线程

（1）创建Callable接口的实现类，并实现call()方法，该call()方法将作为线程执行体，并且有返回值。

```java
public interface Callable
{
   V call() throws Exception;
}
```

（2）创建Callable实现类的实例，使用FutureTask类来包装Callable对象，该FutureTask对象封装了该Callable对象的call()方法的返回值。（FutureTask是一个包装器，它通过接受Callable来创建，它同时实现了Future和Runnable接口。）

（3）使用FutureTask对象作为Thread对象的target创建并启动新线程。

（4）调用FutureTask对象的get()方法来获得子线程执行结束后的返回值

实例代码：

```java
package com.thread;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.FutureTask;

public class CallableThreadTest implements Callable<Integer>
{

    public static void main(String[] args)
    {
        CallableThreadTest ctt = new CallableThreadTest();
        FutureTask<Integer> ft = new FutureTask<>(ctt);
        for(int i = 0;i < 100;i++)
        {
            System.out.println(Thread.currentThread().getName()+" 的循环变量i的值"+i);
            if(i==20)
            {
                new Thread(ft,"有返回值的线程").start();
            }
        }
        try
        {
            System.out.println("子线程的返回值："+ft.get());
        } catch (InterruptedException e)
        {
            e.printStackTrace();
        } catch (ExecutionException e)
        {
            e.printStackTrace();
        }

    }

    @Override
    public Integer call() throws Exception
    {
        int i = 0;
        for(;i<100;i++)
        {
            System.out.println(Thread.currentThread().getName()+" "+i);
        }
        return i;
    }

}
```

二、创建线程的三种方式的对比

1、采用实现Runnable、Callable接口的方式创建多线程时，

优势是：

线程类只是实现了Runnable接口或Callable接口，还可以继承其他类。

在这种方式下，多个线程可以共享同一个target对象，所以非常适合多个相同线程来处理同一份资源的情况，从而可以将CPU、代码和数据分开，形成清晰的模型，较好地体现了面向对象的思想。

劣势是：

编程稍微复杂，如果要访问当前线程，则必须使用Thread.currentThread()方法。

2、使用继承Thread类的方式创建多线程时，

优势是：

编写简单，如果需要访问当前线程，则无需使用Thread.currentThread()方法，直接使用this即可获得当前线程。

劣势是：

线程类已经继承了Thread类，所以不能再继承其他父类。

3、Runnable和Callable的区别

(1) Callable规定（重写）的方法是call()，Runnable规定（重写）的方法是run()。

(2) Callable的任务执行后可返回值，而Runnable的任务是不能返回值的。

(3) call方法可以抛出异常，run方法不可以。

(4) 运行Callable任务可以拿到一个Future对象，表示异步计算的结果。它提供了检查计算是否完成的方法，以等待计算的完成，并检索计算的结果。通过Future对象可以了解任务执行情况，可取消任务的执行，还可获取执行结果。

分类： Java并发编程

标签： java并发编程

« 上一篇： 再谈Spring AOP
» 下一篇： 再谈多线程编程（一）——线程的概念、多线程的创建、守护线程、线程状态的转化

posted @ 2017-12-03 16:07 小淞鼠 阅读(29394) 评论(0) 编辑 收藏