



博客园 首页 新随笔 联系 订阅 管理

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

- 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。
- 创建Thread子类的实例，即创建了线程对象。
- 调用线程对象的start()方法来启动该线程。

```
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

- 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。
- 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。
- 调用线程对象的start()方法来启动该线程。

示例代码如下：

```
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

- 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。

```
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

实例代码如下：

```
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()。

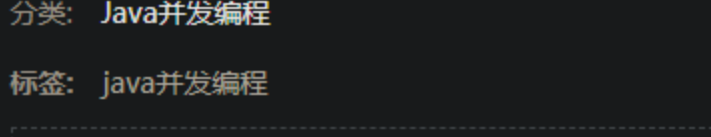
(2) Callable的任务执行后可返回值，而Runnable的任务是不能返回值的。

(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，并检查计算的结果。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



小淼鼠 关注 - 4 粉丝 - 11 4 推荐 0 反对

« 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 小淼鼠 阅读(29394) 评论(0) 编辑 收藏

最新评论 刷新页面 返回顶部

登录后才能发表评论，立即 登录 或 注册， 访问 网站首页

- 【推荐】阿里云实时计算 Flink 训练营重磅开启，4天技能突破，抢天猫精灵！
- 【推荐】阿里云春招即将开始，提前下载面试宝典稳拿Offer
- 【推荐】大型组态、工控、仿真、CAD/GIS 50万行VC++源码免费下载！
- 【推荐】免费领取最高6000元好云礼！15种权益祝你云气爆棚
- 【推荐】阿里云Java训练营，名师带你5天实战Spring Boot 2.5，抢智能音箱
- 【推荐】注册 Amazon Web Services(AWS) 账号，成为博客园赞助商
- 【推荐】华为HMS Core Discovery直播间—七个推送技巧带你玩转App运营

AWS免费产品：
· 如何在AWS上免费构建网站
· AWS免费云存储解决方案
· 在AWS上免费构建数据库
· AWS上的免费机器学习

最新资讯：
· 可再生能源重大突破！液态阳光来了：历经20年研究
· 荣耀渠道商：勒索病毒勒索投入，宁可今年一年不挣钱
· 知乎：盈利之路，道阻且长
· 数字货币挖矿狂潮：矿机售罄、显卡难求
· 这届年轻人，买衣服都不选快时尚了
» 更多新闻...

随笔分类 (29)

DB/SQL(1)
Git(7)
Java Web(2)
Java并发编程(2)
Java基础知识(6)
Linux(1)
测试技术(3)
前端知识(6)
性能测试(1)

最新评论

1. Re:记录一个因sqlmap导致的错误
感谢，找了半天，终于知道问题了！
--vanishke

2. Re:一方包、二方包、三方包是什么？
👍
--郝晓友

阅读排行榜

1. Java String 转 Long 两种方法区别(8948)
2. 用XPath精确定位节点元素&selenium使用XPath定位之完整篇(59049)
3. Java创建线程的三种方式及其对比(29389)
4. Jgit - java实现git操作(28164)
5. 一方包、二方包、三方包是什么？ (14400)
6. Java调用cmd执行maven命令(9374)
7. Linux下ps -ef和ps aux的区别(8610)
8. 初始化一个static的Map变量(5897)
9. 利用selenium webdriver点击alert提示框(5659)
10. nginx结合tomcat一起使用(5482)
11. 如何解决包冲突问题(5089)
12. Java加载jar文件并调用jar文件中带有参数和返回值的类(4518)
13. maven install时自动运行单元测试(4127)
14. 持续集成与灰度发布(4011)
15. 记录一个因sqlmap导致的错误(3575)

推荐排行榜

1. 一方包、二方包、三方包是什么？ (10)
2. Java创建线程的三种方式及其对比(4)
3. 用XPath精确定位节点元素&selenium使用XPath定位之完整篇(3)
4. 一个资深面试官的测试工程师招聘心得 (1)
5. Java String 转 Long 两种方法区别(1)