# 创建多线程的第三种方式

创建多线程共有三种方式：

1）继承 Thread 类，重写 run()方法，run()方法代表线程要执行的任务。

2）实现 Runnable 接口，重写 run()方法，run()方法代表线程要执行的任务。

3）实现 callable 接口，重写 call()方法，call()作为线程的执行体，具有返回值，并且可以对异常进行声明和抛出；使用 start()方法来启动线程

前两种在视频中都有详细介绍，此处将对第三种创建多线程的方式进行介绍：

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。

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。

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。

## 示例代码：

1、实现 Callable 接口，创建线程

```java
package com.imooc.thread;

import java.util.concurrent.Callable;

public class ThirdThread implements Callable<String> {

    @Override
    public String call() throws Exception {

        //方法类型可以根据要返回值的类型进行确认
        String str = "多线程的第三种创建方式";
        return str;
    }

}
```

2、测试线程

```java
package com.imooc.thread;

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

public class ThirdThreadTest {

    //测试方法
    public static void main(String[] args)  {

        Callable<String> call = new ThirdThread();
        FutureTask<String> ft = new FutureTask<>(call);
        Thread t3 = new Thread(ft);

        //启动线程
        t3.start();

        //获取Call方法的返回值：先启动线程才可以获取到Call的返回值
        try {
            System.out.println(ft.get());
        } catch (InterruptedException | ExecutionException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

运行结果：