西南交通大學
Southwest Jiaotong University

移动商务应用开发

第10章 多线程

学习内容

- 什么是多线程?
- 为什么需要多线程?
- 如何创建和启动线程?
- 如何进行多线程间的通信?

什么是多线程?

生活中的例子

- 假设小明有语文、数学、英语3门作业要做。
- 单线程：先做完语文，再做数学，最后做英语。
- 多线程：先做1分钟语文，再做1分钟数学，再做1分钟英语，这样**轮流**做下去。看上去就像同时在做3门作业一样。

什么是多线程?

操作系统的多线程模型

- 当手机上同时运行浏览器、微信和QQ音乐时，操作系统会**轮流**让多个任务交替执行，例如让浏览器执行0.001秒，让微信执行0.001秒，再让QQ音乐执行0.001秒。在人看来，CPU就是在同时执行多个任务。



什么是多线程?

- **线程**是
  - 一个**单一顺序**的控制流
  - 操作系统调度的**最小单元**。
- 同一个线程内的代码是**线性执行**的。
- **多线程**则是并发执行多个线程。
  - 虽然看起来像是“同时”执行多个线程，但事实上只是**轮流让多个线程交替执行**，**每个时刻只执行一个线程**。
- 当多个线程并发运行时，如何调度线程完全由操作系统决定，程序本身无法确定线程的调度顺序。

为什么需要多线程?

- 当一个Android应用启动时，系统会创建一个线程，称**作主线程**，又称**UI线程**。
- 默认所有的任务都在主线程中执行。
- 同一线程内的任务是**线性执行**的，遵循FIFO原则。
  - 因此，一旦前面有耗时的任务未完成，后面的任务都需要等待。这很容易造成“应用无响应”的现象，用户交互体验不好。
- Android应用开发需要**多线程**编程。
  - 把**与用户交互的任务**放在**主线程**中，把**耗时的任务**（如访问网络、查询数据库）放在**其它线程**中。除了主线程外的其它线程，称作**子线程**，又称**工作线程**。

如何创建和启动线程？

- Java语言支持多线程编程。Android应用开发采用相同的语法。
- Java用Thread对象表示一个线程。
  - **创建线程对象**：创建Thread对象时传入一个Runnable对象，并重写run()方法
  - **启动线程**：调用start()方法
- 当创建Thread对象时，系统并没有创建对应的线程，当调用start()时，系统才**真正创建并启用了**对应的线程。
- run()方法内的代码都在**子线程**中执行。
- run()方法执行完成后，线程会自动终止。

如何创建和启动线程?

- 代码展示

```
// 创建线程对象
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        for (int i = 0; i < 100; i++) { // 子线程中运行
            Log.i(TAG, "子线程 " + i);
        }
    }
});

// 启动线程
thread.start();
for (int i = 0; i < 100; i++) { // 主线程中运行
    Log.i(TAG, "主线程 " + i);
}
```

如何创建和启动线程?

- 结果展示

```
15:43:18.113 myTAG    I   主线程 0
15:43:18.113 myTAG    I   主线程 1
15:43:18.113 myTAG    I   主线程 2
15:43:18.113 myTAG    I   子线程 0
15:43:18.113 myTAG    I   子线程 1
15:43:18.113 myTAG    I   主线程 3
15:43:18.113 myTAG    I   子线程 2
15:43:18.113 myTAG    I   主线程 4
```

- 当多个线程并发运行时，如何调度线程完全由操作系统决定，程序本身无法确定线程的调度顺序。

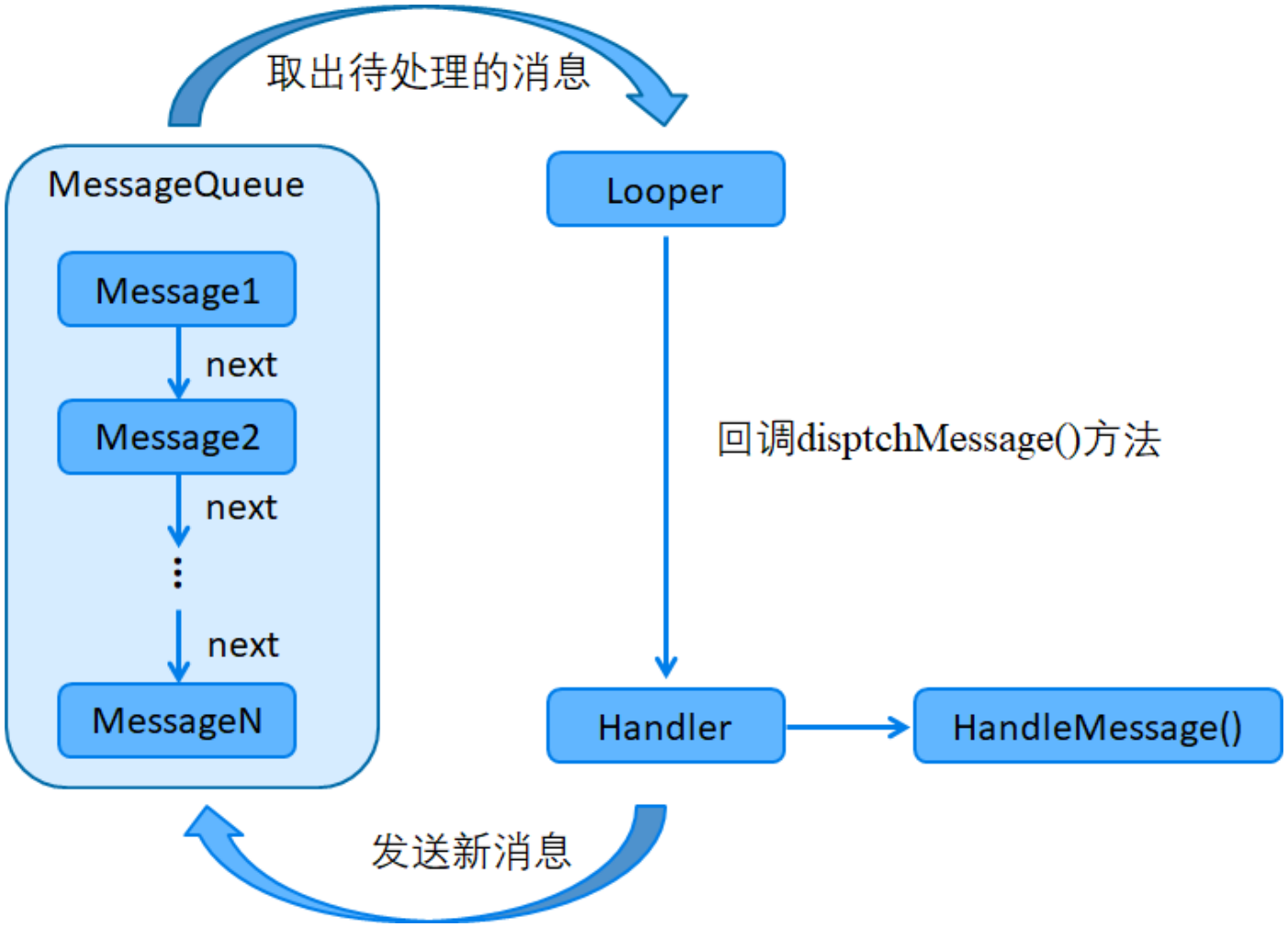
如何进行多线程间的通信

- Android是**线程不安全**的。
  - 多个线程同时更改一个数据可能会造成数据错误。
  - 举例说明：假设售票系统有1000张票，A和B同时来买票，如果是线程不安全，那么可能售票系统可能出现执行两次1000-1的情况，最终结果是A和B都买完后剩下999张票，而不是998张。
- 如何保证线程安全?
- Android不允许在子线程中更新UI，**只能在主线程中更新UI**。
- 如何在子线程中通知主线程更新UI呢?
  - 借助**Handler机制**

如何进行多线程间的通信

- Android提供了**Handler异步消息通信机制**，用于**在不同线程间传递消息**。通过Handler异步消息通信机制，可以实现**多线程间的通信**。
- Handler 异步消息处理机制主要由四个部分组成：
  - Message：消息
  - MessageQueue： 消息队列，存放Handler发送过来的消息
  - Looper：从MessageQueue中获取Message，然后交给Handler处理
  - Handler：消息处理者，发送消息和处理消息

如何进行多线程间的通信



如何进行多线程间的通信

- Looper与MessageQueue
- 一个线程只能有一个Looper对象和一个MessageQueue对象。初始化一个Looper对象时，系统会自动也初始化一个MessageQueue对象。

如何进行多线程间的通信

- Looper与MessageQueue
- 如何为线程创建Looper对象？
  - 主线程：系统已经默认为主线程创建了一个Looper对象，并开启循环读取消息，所以我们无需自行创建。
    - Looper类提供了一个getMainLooper()方法，通过这个方法我们可以在任何地方获取到主线程的Looper，且主线程的Looper不能退出。
  - 子线程：通过调用Looper.prepare()从而为子线程创建一个Looper对象，再调用Looper.loop()开启循环读取消息。

如何进行多线程间的通信

- Handler
- 一个线程允许有多个Handler对象，但通常发送消息和处理消息的是同一个Handler对象。
- 当一个Handler对象被创建时，它会**自动关联**到所在的线程，以及该线程的Looper对象。
- 一个Handler对象可以**在任意线程发送消息**，但它只能**在关联的线程上处理消息。**
  - 我们在主线程创建了一个Handler实例处理消息，那么不管这个Hanlder实例从哪个线程发送消息过来，这些消息最终都会回到主线程的MessageQueue中，然后通过主线程的Looper不断读取消息，再交给主线程的Handler实例来处理。

如何进行多线程间的通信

- Handler
- 如何创建一个Handler对象？
  - 通过调用构造方法new Handler(Looper looper)来创建一个与线程绑定的Handler对象。

```
// 以主线程为例
Handler handler = new Handler(Looper.getMainLooper());
```

如何进行多线程间的通信

- Handler
- 如何通过Handler对象实现子线程和主线程的通信?
  - 方法一：在子线程中通过Handler的sendMessage()发送消息，并在主线程中重写Handler的handleMessage()方法。
    - sendMessage()方法传入的参数是Message对象。
  - 方法二：在子线程中通过调用Handler的post()方法发送消息，并重写该Runnable对象的run()方法，并将更新UI等操作放在run方法内。
    - post()方法传入的参数是一个Runnable对象。

如何进行多线程间的通信

方法一：代码展示

```
// 创建Handler对象
handler = new Handler(getMainLooper()){
    @Override
    public void handleMessage(@NonNull Message msg) {
        super.handleMessage(msg);
        if (msg.what == 1) {
            // 更新UI的操作必须在handler的handleMessage回调中完成
            textView.setText(String.valueOf(num));
        }
    }
};

Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        num += 1;
        // sendMessage方法负责发送消息给handler
        handler.sendEmptyMessage(1);
    }
});
thread.start();
```

如何进行多线程间的通信

方法二：代码展示

```
handler = new Handler(getMainLooper());
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        num += 1;
        handler.post(new Runnable() {
            @Override
            public void run() {
                textView.setText(String.valueOf(num));
            }
        });
    }
});
thread.start();
```

如何进行多线程间的通信

- 拓展：Activity的runOnUiThread()方法
  - runOnUiThread() 用于**从子线程中切换到主线程**来执行一些需要主线程执行的操作，底层调用了**Handler的post()** 。

```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        num += 1;
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                textView.setText(String.valueOf(num));
            }
        });
    }
});
thread.start();
```

学习内容

- **什么是多线程?**
  - 并发执行多个线程，本质上是轮流让多个线程交替执行
- **为什么需要多线程?**
  - 避免应用长时间无反应
- **如何创建和启动线程?**
  - 用Thread对象来表示线程，通过创建Thread对象时传入Runnable对象来构建线程对象，通过调用start()启动线程
- **如何进行多线程间的通信?**
  - Handler 机制