CLASE HANDLER Y RUNNABLE

Asignatura: Programación Multimedia y Dispositivos Móviles

Autores: Pablo Rodríguez Peña

Centro: IES Ramón del Valle Inclán

Curso: 2023/24

ÍNDICE

Manejo de Threads en Android

En Android, existen los procesos o threads, los cuales son una instancia de una aplicación en ejecución. Son administrados por el SO y pueden contener una o varias actividades.

Un hilo es la unidad más pequeña de ejecución dentro de un proceso. Un proceso puede tener varios hilos que se ejecutan de manera concurrente. Estos comparten la misma memoria y recursos dentro de un proceso. Sin embargo, cada hilo tiene su propia pila de ejecución.

Es importante manejar los hilos de manera adecuada para evitar bloquear la interfaz de usuario y proporcionar una experiencia fluida a los usuarios.

Hay dos tipos principales de hilos: el hilo principal (UI thread), el cual se encarga de la interfaz de usuario, y los hilos secundarios (background threads) que se ocupan de otras operaciones en segundo plano.

```
1 usage
Thread miThread = new Thread();
miThread.start();
```

En Java se define un Thread y se inicia su ejecución:

Se puede especificar el código a ejecutar creando una subclase de Thread o implementando la interfaz Runnable:

```
no usages
public class MyThread extends Thread {
    public void run() {
        System.out.println("MyThread running");
    }
}

1 usage
Runnable myRunnable = new Runnable() {
    public void run() {
        System.out.println("Runnable running");
    }
};
1 usage
Thread thread = new Thread(myRunnable);
thread.start();
```

Conceptos

Clase Handler

La biblioteca Android ofrece diversas herramientas para facilitar la gestión de hilos en las aplicaciones. Una de ellas son los Handler.

Permiten enviar y procesar mensajes asociados con un hilo específico y ayudan a evitar el bloqueo de la interfaz de usuario al permitir la ejecución de tareas a la vez.

Transferir tareas a otro proceso

Cuando se trata de transferir trabajo a otro hilo o proceso, como se menciona, se pueden utilizar Message y Runnable para comunicarse entre hilos.

Envío de un Message:

Se utiliza un Message para enviar datos y mensajes entre hilos.

El Handler se encarga de recibir el Message y ejecutar la operación correspondiente.

Este enfoque es útil cuando se necesitan datos más complejos o instrucciones específicas para el hilo receptor.

```
// Envío de un Message desde el hilo emisor
Message message = handler.obtainMessage();
message.what = OPERACION_PERSONALIZADA;
message.obj = datosPersonalizados;
handler.sendMessage(message);
```

Envío de un Runnable:

Los Runnables se utilizan cuando se conocen exactamente los pasos de ejecución que deben realizarse en el hilo receptor.

Encapsulan el código que se ejecutará de forma asíncrona en el hilo receptor.

```
// Envío de un Runnable desde el hilo emisor
new *
Runnable myRunnable = new Runnable() {
    new *
    @Override
    public void run() {
        // Código que se ejecutará de forma asíncrona en el hilo receptor
    }
};

// Ejecutar el Runnable en un hilo específico usando un Handler
handler.post(myRunnable);
```

Combinación con Handler:

La combinación de Message, Runnable, y Handler permite una comunicación y ejecución de tareas asíncronas de manera efectiva.

```java
// Crear un Handler asociado al hilo principal (UI)
Handler handler = new Handler(Looper.getMainLooper());

// Envío de un Message
Message message = handler.obtainMessage();
message.what = OPERACION_PERSONALIZADA;
message.obj = datosPersonalizados;
handler.sendMessage(message);

// Envío de un Runnable
new *
Runnable myRunnable = new Runnable() {
    new *
    @Override
    public void run() {
        // Código que se ejecutará de forma asíncrona en el hilo receptor
    }
};

// Ejecutar el Runnable en un hilo específico usando un Handler
handler.post(myRunnable);
```

Los Handlers gestionan los Runnables haciendo uso de dos componentes asociados al thread, un *MessageQueue* y un *Looper* .

Looper

Looper actúa como un intermediario que toma mensajes de la cola y los envía al componente correcto para su procesamiento, asegurando que las operaciones se realicen de manera ordenada y secuencial.

Un Looper puede tener vinculados uno o varios Handlers. Estos Handlers, pueden ser del mismo productor o de distintos productores.

MessageQueue

Es una lista enlazada ilimitada de mensajes para procesar en el hilo del consumidor. Cada Looper, y Thread, tiene como máximo un MessageQueue.

Cuando se crea un thread, ya sea el thread principal creado por Android para nuestra aplicación o uno que creamos con new Thread(), se crean también su Looper y su MessageQueue.

Cuando se crea un Handler con new Handler(), éste se vincula al MessageQueue y al Looper del contexto en el que se crea

Por tanto, un thread tiene finalmente asociados un Handler, un Looper y una MessageQueue que a su vez tiene asociados Mensajes, siendo el Handler el origen y destino de los mensajes de la cola.

El Looper está continuamente esperando la llegada de trabajos al MessageQueue. Estos trabajos pueden ser Messages o Runnables. Cuando los trabajos llegan, el Looper reacciona en función del tipo de trabajo de que se trate.

En el contexto del manejo de hilos en Android, estas acciones son parte del flujo de trabajo entre productores (o emisores de tareas) y consumidores (o receptores de tareas) de hilos. Veamos cómo estas acciones son realizadas y su objetivo de manera más sencilla:

Insert (Inserción):

Quién: El productor o emisor de la tarea, que podría ser cualquier hilo, utiliza un Handler conectado al hilo consumidor.

Objetivo: Enviar una tarea (Runnable) al hilo consumidor para su ejecución.

```
// Productor (hilo que envía la tarea)
Handler handler = new Handler();
new *
handler.post(new Runnable() {
    new *
    @Override
    public void run() {
        // Tarea a ejecutar en el hilo consumidor
    }
});
```

Retrieve (Recuperación):

Quién: El Looper, que se ejecuta en el hilo consumidor, recupera mensajes de la cola secuencialmente.

Objetivo: Preparar mensajes, incluidos los Runnables, para su ejecución.

```
// Consumidor (hilo que recibe y ejecuta la tarea)
Looper looper = Looper.myLooper();
MessageQueue messageQueue = looper.getQueue();
// El Looper recupera y prepara los mensajes para su ejecución
```

Dispatch (Despacho):

Quién: El Handler asociado al Looper.

Objetivo: Ejecutar la tarea asociada al mensaje recuperado. En el caso de Runnables, esto implica llamar al método run() del Runnable para llevar a cabo la tarea específica en el hilo consumidor.

```java
// Consumidor (hilo que ejecuta la tarea)
Handler handler = new Handler();
new *
handler.post(new Runnable() {
    new *
    @Override
    public void run() {
        // Tarea asociada al mensaje
    }
});
```

Bibliografía

1.- https://umhandroid.momrach.es/handlers/
2.- https://developer.android.com/reference/android/os/Handler
3.- https://www.infor.uva.es/~fdiaz/sd/doc/hilos