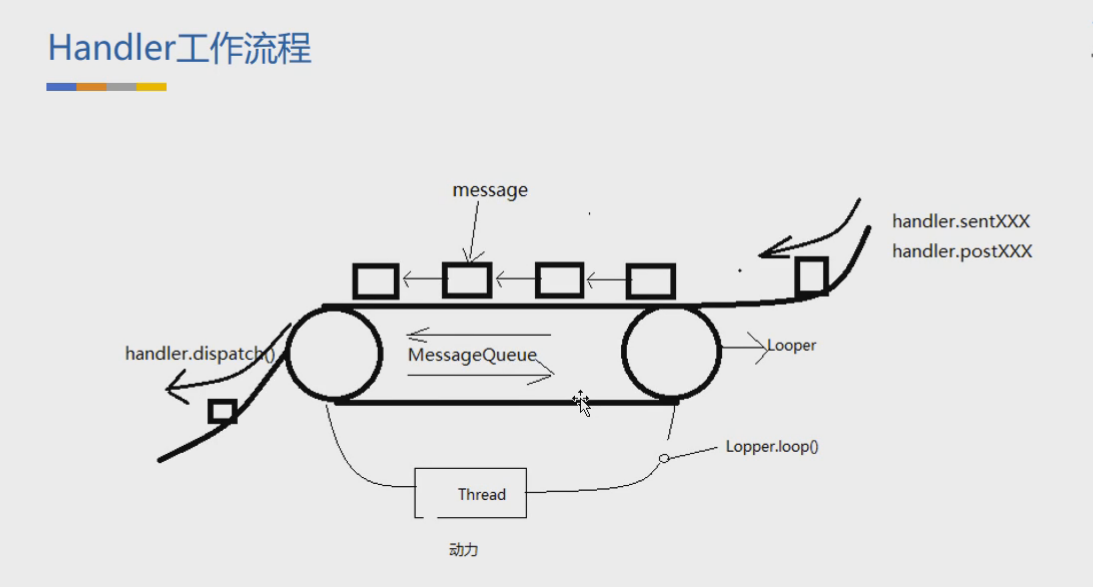
Handler原理

源码epoll 设计思路 设计模式 异步消息和同步消息 消息屏障 HandlerThread IntentServer

消息管理机制

ActivityThread：AMS 围绕handler

Lancher -> zygote ->fork进程 -> 分配jvm -> ActivityThread -> main() -> Looper.prepareMainLooper() -> Looper().loop()



什么时候返回一个msg为空？

应用退出 messagequeue.quit()

Handler -> sendMessage() -> handler.enqueueMessage() -> mQueue.enqueueMessage(队列入队)

-> Looper.loop()循环调用 -> next(取消息) -> handler.dispatchMessage() ->

->handler.handlerMessage()

子线程共享内存到主线程

内存不分线程

mQueue 在子线程还是主线程？

Messagequeue只是个容器，可以主线程或子线程创建



消息队列 ：单链表实现的优先级队列

Message -> next ->Message -> next -> Message

队列的先后顺序，如何实现的？

根据时间排序。当有消息进入队列时，轮询判断所有消息执行的时间并排序。

排序算法： 插入排序

队列：先进先出表

入队不能保证，出队根据时间进行优先级排序。

Looper 源码

核心：构造函数，prepare (), loop() , ThreadLocal成员变量

Looper 初始化

Looper.prepare() 中调用私有构造函数

Looper.prepare() 中会先 threadLocalMap.get() 获取判空

1.为空 创建sThreadLocal.set(new Looper())

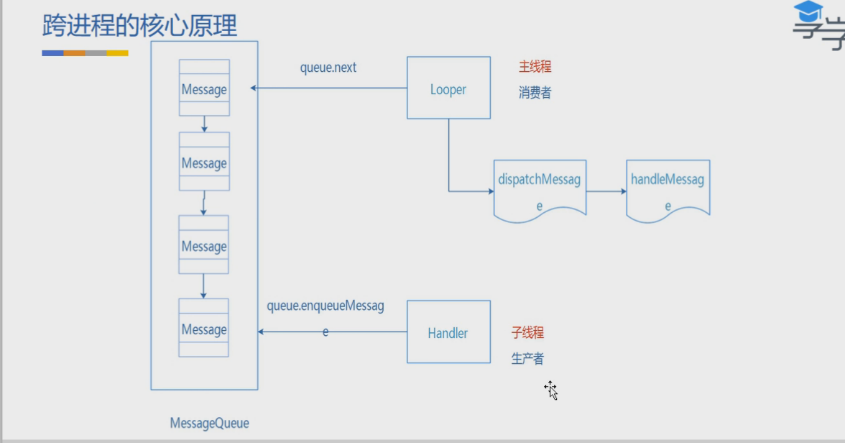
2.不为空 返回异常 一个线程只能创建一个looper

Looper.loop() 开始循环

ThreadLocalMap为ThreadLocal内部类

Set值时获取当前线程

消息的唤醒和睡眠机制



生产消费者的设计模式

nativePollOnce 休眠

nativeWake 唤醒

Epoll机制

子线程quit工作：

唤醒线程 -> messagequeue -> null -> 退出

一个线程只有一个可以操作MessageQueue的地方

入队和出队互斥

面试题

1.为什么分配多个jvm？

每个应用拥有独立空间，不影响其他应用。

2.主线程可不可以调用Looper中的quit方法？

主线程不能调用quit方法

3.两个方面阻塞 ？

1．Message不到时间，自动唤醒

2．队列为空时阻塞，无限等待，添加消息时唤醒

4.MessageQueue中的队列阻塞了什么？

线程。

5. Synchronized 为什么叫内置锁？

加锁 释放锁 JVM内部实现

分为：类锁 对象锁

功能：可以修饰方法 修饰代码块

6.ThreadLocal 用途？

ThreadLocal 线程上下文的存储变量

多线程共享变量私有化，每个线程都持有变量副本。

7.一个线程一个Looper如何保证？

Looper中含有static final ThreadLocal<Looper>的成员变量，当执行Looper.prepare时，会通过这个threadlocal成员变量get值，当为空时，调用threadLocal.set（new Looper()）方法， 会对当前线程的threadLocals成员变量进行赋值操作，threadLocals为ThreadLocalMap对象，ThreadLocalMap为ThreadLocal的内部静态类，内部包含Entry类型的数组，Entry的key为当前threadlocal对象，value为looper对象。Entry继承自WeakReference<ThreadLocal>

当不为空时，抛出“Only one Looper may be created per thread”异常。

8.为什么Hanlder会提示内存泄漏，建议使用static？

内部类持有外部类引用？为什么指有handler会存在内存泄漏的问题？

Handler.enqueueMessage 中设置msg.target = this，将当前handler赋值给target，Message post 20m 持有handler-> handler持有activity -> Activity