00 技能树

- Java
  - 基本特性
    - 多态
    - 继承
    - 封装
  - 集合类
    - List
      - Vector
        - 原理
        - 区别
        - 使用场景
      - ArrayList
        - 原理
        - 区别
        - 使用场景
        - ArrayList什么时候指定数组大小为多少
        - ArrayList如何进行扩容
        - ArrayList在1.7和1.8有什么区别
        - ArrayList为什么新增和删除效率很慢
      - LinkedList
        - 原理
        - 区别
        - 使用场景
    - Set
      - HashSet
        - 原理
        - 区别
        - 使用场景
      - LinkedHashSet
        - 原理
        - 区别
        - 使用场景
      - TreeSet

        - 原理
        - 区别
        - 使用场景
    - Map
        - HashMap
            - 原理
            - 区别
            - 使用场景
            - 什么是头插法和尾插法
            - 头插法会导致什么问题
            - HashMap什么时候进行扩容
            - HashMap如何进行扩容
            - 如何计算hash
            - HashMap数组的初始值是多少
            - 为什么HashMap的初始值是16
            - 为什么重新equals方法需要重新hashCode
            - HashMap的put流程
            - HashMap的get流程
            - Collections.synchronizedMap如何实现线程安全
            - HashMap与HashTable的区别
            - hash表是一个怎样的数据结构
            - 如何避免hash碰撞
            - hash表容量有什么特点
        - TreeMap
            - 原理
            - 区别
            - 使用场景
        - HashTable
            - 原理
            - 区别
            - 使用场景
            - 为什么HashTable效率低
            - 为什么HashTable不允许key或者value为空
            - HashTable初始化容量是多少
            -
- 多线程

- 进程与线程
  - 什么是进程
  - 什么是线程
  - 进程与线程的区别
- 多线程入门类和接口
  - Thread
    - Thread是什么的实现类
    - yield方法实际意义
    - join方法实际意义
    - sleep方法实际意义
  - Runable
    - Thread类与Runnable比较
  - Callable
    - 与Runnable的区别
    - 基本使用
  - Future
    - cancel方法实际意义
  - FutureTask
    - RunnableFuture啥玩意
- 线程组
  - Thread可以独立于ThreadGroup存在吗
  - 如果新建线程没有指定线程组该线程属于哪个线程组
  - 线程组有什么作用
- 线程的优先级
  - 线程优先级可以在实际业务中使用确定优先级吗
  - 线程组内某个线程优先级大于线程组优先级最终优先级是什么
- 线程状态
  - 线程有哪些状态
  - NEW
    - 反复调用同一个线程的start方法是否可行
    - 假如一个线程执行完毕处于TERMINATED，是否可以再次调用start方法
  - RUNNABLE
  - BLOCKED
  - WAITTING
  - TIMED_WAITING
  - TERMINATED

- 线程间通信
  - 锁与同步
  - wait/notify机制
  - 信号量
    - volatile
  - 管道
  - 其他通信方式
    - join方法
    - sleep方法
    - ThreadLocal
    - InheritedThreadLocal
- volatile
  - volatile作用什么
  - volatile原理
- synchronized
  - synchronized修饰在方法上与修饰代码块区别
  - synchronized修饰静态方法与修饰类区别
  - synchronized原理
  - synchronized属于什么锁
  - synchronize的不足
- 锁状态
  - 一个对象有几种锁状态
  - 一个对象的锁存放在什么地方
  - 对象头存放什么信息
  - 偏向锁原理
  - 轻量级锁原理
  - 重量级锁原理
- 锁的分类
  - 什么是乐观锁和悲观锁
  - 什么是可重入锁不可重入锁
  - 什么是公平锁和非公平锁
  - 什么是读写锁和排他锁
  - 非公平锁会导致什么问题
- CAS
  - 什么是CAS
  - CAS原理

        - CAS实现原子操作会导致那些问题
        - 如何解决CAS实现原子操作导致的问题
    - 线程池
        - 线程池构造参数及其含义
        - 常见的阻塞队列有哪些
        - 阻塞队列底层原理
        - 线程池状态有哪些
        - 线程池工作流程
        - 线程池如何实现线程复用
        - 常见的线程池有哪些
    - 同步容器和并发容器
        - 线程安全的容器类有哪些
        - Vector如何实现线程安全
        - HashTable如何实现线程安全
        - ConcurrentHashMap底层数据结构
        - ConcurrentHashMap如何实现线程安全
        - CopyOnWriteArrayList的原理
        - CopyOnWriteArrayList的优点
        - CopyOnWriteArrayList的缺点
        - 什么是工作窃取算法
- JVM
    - 内存模型
    - 堆栈的区别
    - 类的生命周期
    - 类加载流程
    - 什么是双亲委托
    - 如何打破双亲委托机制
    - 怎么判断一个对象是否可回收
    - 引用类型有哪些
    - 垃圾回收算法
    - Android垃圾回收与JVM的区别
- 注解
    - 原理
- 反射
    - 原理
    - 为什么性能不好

        - 反射线程安全吗
        - 反射是如何保证线程安全的
    - 泛型
        - 原理
        - 什么是类型擦除
        - 什么是限定通配符和非限定通配符
        - Array可以使用泛型吗
- Android
    - 基础
        - 四大组件
            - Activity
                - activity的启动模式与taskA...关键字
                - 横竖屏切换activity生命周期
                - 跳转透明activity的生命周期
                - 锁屏和解锁的生命周期
                - 弹出dialog生命周期
                - onStop方法一定执行吗
                - Intent传递数据大小是否有限制
                - 如何在activity间传递大量数据
                - activity的newIntent什么时候会执行
                - 显示启动和隐式启动的区别
                - onCreate和onRestoreInstance方法中恢复数据的区别
                - activity间传递数据的方式有哪些
                - 跨App启动Activity的方式
                - App被kill后如何恢复数据
            - Service
                - service生命周期
                - start与bind的区别
                - Service如何与Activity通信
                - IntentService作用与原理
            - BroadcastReceiver
            - ContentProvider
                - 原理
        - 基本View
            - View
                - view绘制流程

        - 自定义View的wrap_content不起作用的原因
        - 为什么onCrete获取不到View的宽高
        - View.post与Handler.post的区别
        - 自定义View的流程及注意事项
        - 自定义View如何考虑机型适配
        - 自定义控件优化方案
        - View事件分发机制
        - View的刷新机制
        - 如何实现点击事件被拦截但是想传到下面的view
        - 如何解决view的事件冲突
        - 子线程一定不能更新UI吗
    - ViewGroup
        - LinearLayout、 FrameLayout 和 RelativeLayout 哪个效率高
    - Fragment
        - Fragment生命周期
        - activity与fragment的通信方式有哪些
        - FragmentPageAdapter与FragmentStatePageAdapter的区别
        - Fragment如何实现懒加载
        - Fragment的嵌套问题如何处理
    - RecycleView
        - RecyclerView缓存机制
        - RecyclerView滑动回收复用机制
        - RecyclerView的刷新回收复用机制
        - RecyclerView 为什么要预布局
        - RecyclerView 的性能优化
        - RecyclerView 吸附效果如何实现
        - RecyclerView 实现瀑布流的原理
        - RecycleView绘制流程
        - RecycleView局部刷新原理
        - RecycleView点击事件原理
    - ListView
        - 缓存原理
        - 与RecyclerView的区别
    - Bitmap
        - 如何计算bitmap的内存占用
        - bitmap的压缩方式

        - LruCache与DiskLrucher原理
        - 如何设计一个图片加载库
        - 如何加载一张100M的图片
    - WebView
        - 如何优化加载速度
        - 漏洞
        - JsBridge原理
    - Toast
        - Toast原理
            - AIDL
        - Toast显示流程
            - makeToast创建Toast对象主要创建Toast的View
            - 执行show方法
            - AIDL转发给NMS处理
                - NMS运行在SystemServer进程
            - NMS为Toast创建一个token并添加到WMS
            - WMS调用addToken
            - NMW回调token给Toast
            - App进程的Toast通过Handler切换到主线程执行WM.addView显示在window
        - Toast消失流程
            - 当显示时间到达后
            - NMS回调TN.hide方法进而由App进程调用WM移除Toast
        - 为什么不能在子线程显示Toast
            - Toast.TN.Handler
        - 如何在子线程显示Toast
            - Looper
        - 如何自定义实现一个Toast
- SharedPerferences
    - SP底层原理
    - SP.commit与SP.apply的区别
    - SP.apply流程8.0之前
        - 将数据更新到内存
        - 创建等待任务内部等待写入任务完成
        - 将等待任务加入到列表
        - 创建写入任务执行等待任务

        - 移除等待任务
        - 将写入任务加入队列中
        - 通知回调
    - SP如何导致ANR现象
        - ActivityThread回调组件的一些生命周期时候
            - Activity.onPause
            - Activity.onStop
        - 会调用QueueWork.waitFinish去等待所有的写入任务执行完毕
        - 然而这是个等待任务列表并不是真正的写任务列表
        - 所以真实情况是阻塞主线程等待写任务完成
        - 一些低端机或者apply提交的任务比较多或者apply提交的数据比较大就会anr
    - 如何解决SP导致的ANR
    - SP在android8.0版本做了哪些优化
        - 8.0以前是线程池
        - 8.0之后是HandlerThread
        - 8.0之前QueueWork.waitToFinish执行的是等待队列
        - 8.0之后QueueWork.waitToFinish执行的是写任务队列
        - 8.0之前是全写入
        - 8.0之后只写入最新任务
- 集合类
    - ArrayMap
- 动画
    - 动画的类型及使用场景
- 异常
    - Error和异常有什么区别
    - 怎么防止程序崩溃
    - native崩溃捕获原理
    - 如果已经到了Thread .UncaughtExceptionHandler是否能让程序继续运行
    - ANR发生的原理
    - 如何排查ANR
    - 如何处理页面内存占用过大的问题
- 多线程
    - Handler
        - 原理
        - 主线程为什么不会因为Handler里面的死循环卡死

        - looper如何区别多个handler
        - handler如何实现线程切换的
        - MessageQueue.next原理
        - 什么是同步屏障
        - 同步屏障的作用
        - 为什么handler死循环不会阻塞主线程
        - 为什么死循环不会导致界面卡顿
        - Looper.prepare做了什么事情
        - ThreadLocal含义
        - ThreadLocal内部实现原理
      - HandlerThread
        - 原理
        - 使用场景
      - IntentService
        - 原理
        - 使用场景
      - AsyncTask
        - 原理
- 项目结构
  - MVC及优缺点
  - MVP及优缺点
  - MVVM优缺点
- 进阶
  - JetPack
    - LifeCircle
      - 基本组成
      - 原理
      - Activity如何实现LifeCircle
      - 为什么Activity使用LifeCircle需要添加ReportFragment
      - 如果延迟一段时间注册观察者可以收到注册之前的生命周期回调吗，为什么
      - 如果打开app立马按home键生命周期变化是怎样的
    - DataBinding
  - 热门技术
    - 组件化
      - 什么是模块化

        - 什么是组件化
        - 组件化优缺点
        - 组件间通信
        - ARouter原理
    - 插件化
        - 插件化原理
        - 插件化分类
        - 插件化类加载原理
        - 插件化资源加载原理
    - 热修复
        - 同上
- 第三方框架
    - OkHttp
        - 有哪些拦截器，作用是什么
        - 连接池原理
        - 缓存原理
        - 同步任务流程
        - 异步任务流程
        - 用到了哪些设计模式
        - 如何实现下载进度监听
        - 如何实现上传进度监听
        - 如何实现分段上传
        - 如何实现分段下载
        - 如何实现数据加密传输
        - 如何实现自定义缓存
        - 如果判断当前网速
        - 弱网如何优化
        - 有几个队列，作用是什么
        - 如何取消一个请求
        - Socket底层原理
    - Retrofit
        - 原理
        - 工作流程
        - 设计模式
    - EventBus
        - 原理

        - Glide
            - 原理
            - 缓存机制
            - 图片优化
        - RxJava
            - 原理
        - LeakCanary
            - 原理
            - 为什么新版本不需要在Application中注册了
- FrameWork
    - 虚拟机
        - ART
        - Dalvik
    - 启动流程
        - Android启动流程
        - Activity启动流程
        - Window添加流程
        - APK安装流程
        - 启动一个空的APP会启动几个线程
    - IPC
        - Binder
            - 什么是binder
            - binder原理
            - 一次拷贝原理
            - binder传输数据流程
        - 管道
        - Socket
        - 信号
        - 共享内存
    - WMS
    - AMS
- 性能优化
    - 内存优化
    - 启动优化
    - 布局优化
    - 卡顿优化

        - 网络优化
    - Android个版本差异及特性
- Kotin
    - 不熟
- Flutter
    - 基本原理
    - 绘制流程
    - Widget树
    - Element树
    - RenderObject树
- 设计模式
    - 创建型模式
        - 单例模式
            - 有多少种单例模式
            - 枚举算不算单例
            - 为什么不推荐使用枚举
            - 单例模式不用volatile有什么问题
            - sync关键字为什么放在代码块
            - sync关键字锁的是什么
            - sync关键字放在方法上与代码块有什么区别
            - sync关键字放在静态方法和普通方法有什么区别
            - sync底层实现是什么
            - sync有什么作用
            - sync与volatile的区别
        - 原型模式
        - 简单工厂模式
        - 工厂方法模式
        - 抽象工厂模式
        - 建造者模式
    - 结构型模式
        - 代理模式
        - 适配器模式
        - 桥接模式
        - 装饰器模式
        - 外观模式
        - 享元模式

        - 组合模式
    - 行为型模式
        - 模板方法模式
        - 策略模式
        - 命令模式
        - 责任链模式
        - 状态模式
        - 观察者模式
        - 中介模式
        - 迭代器模式
        - 访问者模式
        - 备忘录模式
        - 解释器模式
- 数据结构
    - 数组
    - 链表
    - 队列
    - 栈
    - 树
    - 图
- 经典算法
    - 排序算法
        - 第一梯队
            - 冒泡排序
            - 选择排序
            - 插入排序
        - 第二梯队
            - 快速排序
            - 希尔排序
            - 归并排序
            - 堆排序
        - 第三梯队
            - 基数排序
            - 桶排序
            - 计数排序
    - 查找算法

- 计算机网络
  - 参考模型
    - OSI七层参考模式
    - TCP/IP五层参考模型
    - TCP/IP四层参考模型
  - 协议
    - HTTP
    - HTTPS
    - DNS
    - TCP
    - UDP

以上内容整理于 幕布文档