原创者：黄钦建（大黄）

ThreadLocal(线程变量副本)

Synchronized实现内存共享，ThreadLocal为每个线程维护一个本地变量。

采用空间换时间，它用于线程间的数据隔离，为每一个使用该变量的线程提供一个副本，每个线程都可以独立地改变自己的副本，而不会和其他线程的副本冲突。

ThreadLocal类中维护一个Map，用于存储每一个线程的变量副本，Map中元素的键为线程对象，而值为对应线程的变量副本。

ThreadLocal在Spring中发挥着巨大的作用，在管理Request作用域中的Bean、事务管理、任务调度、AOP等模块都出现了它的身影。

Spring中绝大部分Bean都可以声明成Singleton作用域，采用ThreadLocal进行封装，因此有状态的Bean就能够以singleton的方式在多线程中正常工作了。

友情链接：深入研究java.lang.ThreadLocal类

Java内存模型：

Java虚拟机规范中将Java运行时数据分为六种。

1.程序计数器：是一个数据结构，用于保存当前正常执行的程序的内存地址。Java虚拟机的多线程就是通过线程轮流切换并分配处理器时间来实现的，为了线程切换后能恢复到正确的位置，每条线程都需要一个独立的程序计数器，互不影响，该区域为“线程私有”。

2.Java虚拟机栈：线程私有的，与线程生命周期相同，用于存储局部变量表，操作栈，方法返回值。局部变量表放着基本数据类型，还有对象的引用。

3.本地方法栈：跟虚拟机栈很像，不过它是为虚拟机使用到的Native方法服务。

4.Java堆：所有线程共享的一块内存区域，对象实例几乎都在这分配内存。

5.方法区：各个线程共享的区域，储存虚拟机加载的类信息，常量，静态变量，编译后的代码。

6.运行时常量池：代表运行时每个class文件中的常量表。包括几种常量：编译时的数字常量、方法或者域的引用。

友情链接： Java中JVM虚拟机详解

“你能不能谈谈，java GC是在什么时候，对什么东西，做了什么事情？”

在什么时候：

1.新生代有一个Eden区和两个survivor区，首先将对象放入Eden区，如果空间不足就向其中的一个survivor区上放，如果仍然放不下就会引发一次发生在新生代的minor GC，将存活的对象放入另一个survivor区中，然后清空Eden和之前的那个survivor区的内存。在某次GC过程中，如果发现仍然又放不下的对象，就将这些对象放入老年代内存里去。

2.大对象以及长期存活的对象直接进入老年区。

3.当每次执行minor GC的时候应该对要晋升到老年代的对象进行分析，如果这些马上要到老年区的老年对象的大小超过了老年区的剩余大小，那么执行一次Full GC以尽可能地获得老年区的空间。

对什么东西：从GC Roots搜索不到，而且经过一次标记清理之后仍没有复活的对象。

做什么： 新生代：复制清理； 老年代：标记-清除和标记-压缩算法； 永久代：存放Java中的类和加载类的类加载器本身。

GC Roots都有哪些：

1. 虚拟机栈中的引用的对象
2. 方法区中静态属性引用的对象，常量引用的对象
3. 本地方法栈中JNI（即一般说的Native方法）引用的对象。

友情链接：Java GC的那些事（上）

友情链接：Java GC的那些事（下）

友情链接：CMS垃圾收集器介绍

Synchronized 与Lock都是可重入锁，同一个线程再次进入同步代码的时候.可以使用自己已经获取到的锁。

Synchronized是悲观锁机制，独占锁。而Locks.ReentrantLock是，每次不加锁而是假设没有冲突而去完成某项操作，如果因为冲突失败就重试，直到成功为止。 ReentrantLock适用场景

1. 某个线程在等待一个锁的控制权的这段时间需要中断
2. 需要分开处理一些wait-notify，ReentrantLock里面的Condition应用，能够控制notify哪个线程，锁可以绑定多个条件。
3. 具有公平锁功能，每个到来的线程都将排队等候。

友情链接： Synchronized关键字、Lock，并解释它们之间的区别

StringBuffer是线程安全的，每次操作字符串，String会生成一个新的对象，而StringBuffer不会；StringBuilder是非线程安全的

友情链接：String、StringBuffer与StringBuilder之间区别

---

fail-fast：机制是java集合(Collection)中的一种错误机制。当多个线程对同一个集合的内容进行操作时，就可能会产生fail-fast事件。 例如：当某一个线程A通过iterator去遍历某集合的过程中，若该集合的内容被其他线程所改变了；那么线程A访问集合时，就会抛出ConcurrentModificationException异常，产生fail-fast事件

---

happens-before:如果两个操作之间具有happens-before 关系，那么前一个操作的结果就会对后面一个操作可见。 1.程序顺序规则：一个线程中的每个操作，happens- before 于该线程中的任意后续操作。 2.监视器锁规则：对一个监视器锁的解锁，happens- before 于随后对这个监视器锁的加锁。 3.volatile变量规则：对一个volatile域的写，happens- before于任意后续对这个volatile域的读。 4.传递性：如果A happens- before B，且B happens- before C，那么A happens- before C。 5.线程启动规则：Thread对象的start()方法happens- before于此线程的每一个动作。

Volatile和Synchronized四个不同点： 1 粒度不同，前者针对变量 ，后者锁对象和类 2 syn阻塞，volatile线程不阻塞 3 syn保证三大特性，volatile不保证原子性 4 syn编译器优化，volatile不优化 volatile具备两种特性：

1. 保证此变量对所有线程的可见性，指一条线程修改了这个变量的值，新值对于其他线程来说是可见的，但并不是多线程安全的。
2. 禁止指令重排序优化。 Volatile如何保证内存可见性: 1.当写一个volatile变量时，JMM会把该线程对应的本地内存中的共享变量刷新到主内存。 2.当读一个volatile变量时，JMM会把该线程对应的本地内存置为无效。线程接下来将从主内存中读取共享变量。

同步：就是一个任务的完成需要依赖另外一个任务，只有等待被依赖的任务完成后，依赖任务才能完成。 异步：不需要等待被依赖的任务完成，只是通知被依赖的任务要完成什么工作，只要自己任务完成了就算完成了，被依赖的任务是否完成会通知回来。（异步的特点就是通知）。 打电话和发短信来比喻同步和异步操作。 阻塞：CPU停下来等一个慢的操作完成以后，才会接着完成其他的工作。 非阻塞：非阻塞就是在这个慢的执行时，CPU去做其他工作，等这个慢的完成后，CPU才会接着完成后续的操作。 非阻塞会造成线程切换增加，增加CPU的使用时间能不能补偿系统的切换成本需要考虑。

友情链接：Java并发编程之volatile关键字解析

CAS（Compare And Swap） 无锁算法： CAS是乐观锁技术，当多个线程尝试使用CAS同时更新同一个变量时，只有其中一个线程能更新变量的值，而其它线程都失败，失败的线程并不会被挂起，而是被告知这次竞争中失败，并可以再次尝试。CAS有3个操作数，内存值V，旧的预期值A，要修改的新值B。当且仅当预期值A和内存值V相同时，将内存值V修改为B，否则什么都不做。

友情链接：非阻塞同步算法与CAS(Compare and Swap)无锁算法

线程池的作用： 在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线程 第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。 第二：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。 第三：提高线程的可管理性。 常用线程池：ExecutorService 是主要的实现类，其中常用的有 Executors.newSingleThreadPool(),newFixedThreadPool(),newcachedTheadPool(),newScheduledThreadPool()。

友情链接：线程池原理

友情链接：线程池原理解析

类加载器工作机制： 1.装载：将Java二进制代码导入jvm中，生成Class文件。 2.连接：a）校验：检查载入Class文件数据的正确性 b）准备：给类的

静态变量分配存储空间 c）解析：将符号引用转成直接引用 3：初始化：对类的静态变量，静态方法和静态代码块执行初始化工作。

双亲委派模型：类加载器收到类加载请求，首先将请求委派给父类加载器完成 用户自定义加载器->应用程序加载器->扩展类加载器->启动类加载器。

友情链接：深入理解Java虚拟机笔记---双亲委派模型

友情链接：JVM类加载的那些事

友情链接：JVM（1）：Java 类的加载机制

一致性哈希：

Memcahed缓存： 数据结构：key,value对 使用方法：get,put等方法

友情链接：hashcode(),equal()方法深入解析

Redis数据结构: String—字符串（key-value 类型） Hash—字典(hashmap) Redis的哈希结构可以使你像在数据库中更新一个属性一样只修改某一项属性值 List—列表 实现消息队列 Set—集合 利用唯一性 Sorted Set—有序集合 可以进行排序 可以实现数据持久化

友情链接： Spring + Redis 实现数据的缓存

---

java自动装箱拆箱深入剖析

谈谈Java反射机制

如何写一个不可变类?

索引：B+，B-,全文索引 Mysql的索引是一个数据结构，旨在使数据库高效的查找数据。 常用的数据结构是B+Tree，每个叶子节点不但存放了索引键的相关信息还增加了指向相邻叶子节点的指针，这样就形成了带有顺序访问指针的B+Tree，做这个优化的目的是提高不同区间访问的性能。 什么时候使用索引：

1. 经常出现在group by,order by和distinc关键字后面的字段
2. 经常与其他表进行连接的表，在连接字段上应该建立索引
3. 经常出现在Where子句中的字段
4. 经常出现用作查询选择的字段

友情链接：MySQL：InnoDB存储引擎的B+树索引算法

友情链接：MySQL索引背后的数据结构及算法原理

Spring IOC （控制反转，依赖注入）

Spring支持三种依赖注入方式，分别是属性（Setter方法）注入，构造注入和接口注入。

在Spring中，那些组成应用的主体及由Spring IOC容器所管理的对象被称之为Bean。

Spring的IOC容器通过反射的机制实例化Bean并建立Bean之间的依赖关系。 简单地讲，Bean就是由Spring IOC容器初始化、装配及被管理的对象。获取Bean对象的过程，首先通过Resource加载配置文件并启动IOC容器，然后通过getBean方法获取bean对象，就可以调用他的方法。 Spring Bean的作用域： Singleton：Spring IOC容器中只有一个共享的Bean实例，一般都是Singleton作用域。 Prototype：每一个请求，会产生一个新的Bean实例。 Request：每一次http请求会产生一个新的Bean实例。

友情链接： Spring框架IOC容器和AOP解析

友情链接：浅谈Spring框架注解的用法分析

友情链接：关于Spring的69个面试问答——终极列表

代理的共有优点：业务类只需要关注业务逻辑本身，保证了业务类的重用性。 Java静态代理： 代理对象和目标对象实现了相同的接口，目标对象作为代理对象的一个属性，具体接口实现中，代理对象可以在调用目标对象相应方法前后加上其他业务处理逻辑。 缺点：一个代理类只能代理一个业务类。如果业务类增加方法时，相应的代理类也要增加方法。 Java动态代理： Java动态代理是写一个类实现InvocationHandler接口，重写Invoke方法，

在Invoke方法可以进行增强处理的逻辑的编写，这个公共代理类在运行的时候才能明确自己要代理的对象，同时可以实现该被代理类的方法的实现，然后在实现类方法的时候可以进行增强处理。 实际上：代理对象的方法 = 增强处理 + 被代理对象的方法

JDK和CGLIB生成动态代理类的区别： JDK动态代理只能针对实现了接口的类生成代理（实例化一个类）。此时代理对象和目标对象实现了相同的接口，目标对象作为代理对象的一个属性，具体接口实现中，可以在调用目标对象相应方法前后加上其他业务处理逻辑 CGLIB是针对类实现代理，主要是对指定的类生成一个子类（没有实例化一个类），覆盖其中的方法 。 Spring AOP应用场景 性能检测，访问控制，日志管理，事务等。 默认的策略是如果目标类实现接口，则使用JDK动态代理技术，如果目标对象没有实现接口，则默认会采用CGLIB代理

SpringMVC运行原理

1. 客户端请求提交到DispatcherServlet
2. 由DispatcherServlet控制器查询HandlerMapping，找到并分发到指定的Controller中。
3. Controller调用业务逻辑处理后，返回ModelAndView
4. DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图
5. 视图负责将结果显示到客户端

友情链接：Spring：基于注解的Spring MVC（上）

友情链接： Spring：基于注解的Spring MVC（下）

友情链接：SpringMVC与Struts2区别与比较总结

友情链接：SpringMVC与Struts2的对比

一个Http请求 DNS域名解析 --> 发起TCP的三次握手 --> 建立TCP连接后发起http请求 --> 服务器响应http请求，浏览器得到html代码 --> 浏览器解析html代码，并请求html代码中的资源（如javascript、css、图片等） --> 浏览器对页面进行渲染呈现给用户

设计存储海量数据的存储系统：设计一个叫“中间层”的一个逻辑层，在这个层，将数据库的海量数据抓出来，做成缓存，运行在服务器的内存中，同理，当有新的数据到来，也先做成缓存，再想办法，持久化到数据库中，这是一个简单的思路。主要的步骤是负载均衡，将不同用户的请求分发到不同的处理节点上，然后先存入缓存，定时向主数据库更新数据。读写的过程采用类似乐观锁的机制，可以一直读（在写数据的时候也可以），但是每次读的时候会有个版本的标记，如果本次读的版本低于缓存的版本，会重新读数据，这样的情况并不多，可以忍受。

友情链接： HTTP与HTTPS的区别

友情链接： HTTPS 为什么更安全，先看这些

友情链接： HTTP请求报文和HTTP响应报文

友情链接： HTTP 请求方式: GET和POST的比较

Session与Cookie：Cookie可以让服务端跟踪每个客户端的访问，但是每次客户端的访问都必须传回这些Cookie，如果Cookie很多，则无形的增加了客户端与服务端的数据传输量， 而Session则很好地解决了这个问题，同一个客户端每次和服务端交互时，将数据存储通过Session到服务端，不需要每次都传回所有的Cookie值，而是传回一个ID，每个客户端第一次访问服务器生成的唯一的ID，客户端只要传回这个ID就行了，这个ID通常为NAME为JSESSIONID的一个Cookie。这样服务端就可以通过这个ID，来将存储到服务端的KV值取出了。 Session和Cookie的超时问题，Cookie的安全问题

分布式Session框架

1. 配置服务器，Zookeeper集群管理服务器可以统一管理所有服务器的配置文件
2. 共享这些Session存储在一个分布式缓存中，可以随时写入和读取，而且性能要很好，如Memcache，Tair。
3. 封装一个类继承自HttpSession，将Session存入到这个类中然后再存入分布式缓存中
4. 由于Cookie不能跨域访问，要实现Session同步，要同步SessionID写到不同域名下。

适配器模式：将一个接口适配到另一个接口，Java I/O中InputStreamReader将Reader类适配到InputStream，从而实现了字节流到字符流的准换。 装饰者模式：保持原来的接口，增强原来有的功能。 FileInputStream 实现了InputStream的所有接口，BufferedInputStreams继承自FileInputStream是具体的装饰器实现者，将InputStream读取的内容保存在内存中，而提高读取的性能。

Spring事务配置方法：

1. 切点信息，用于定位实施事物切面的业务类方法

2. 控制事务行为的事务属性，这些属性包括事物隔离级别，事务传播行为，超时时间，回滚规则。 Spring通过aop/tx Schema 命名空间和@Transaction注解技术来进行声明式事物配置。

Mybatis 每一个Mybatis的应用程序都以一个SqlSessionFactory对象的实例为核心。首先用字节流通过Resource将配置文件读入，然后通过SqlSessionFactoryBuilder().build方法创建SqlSessionFactory，然后再通过SqlSessionFactory.openSession()方法创建一个SqlSession为每一个数据库事务服务。 经历了Mybatis初始化 -->创建SqlSession -->运行SQL语句，返回结果三个过程

Servlet和Filter的区别： 整的流程是：Filter对用户请求进行预处理，接着将请求交给Servlet进行处理并生成响应，最后Filter再对服务器响应进行后处理。

Filter有如下几个用处： Filter可以进行对特定的url请求和相应做预处理和后处理。 在HttpServletRequest到达Servlet之前，拦截客户的HttpServletRequest。 根据需要检查HttpServletRequest，也可以修改HttpServletRequest头和数据。 在HttpServletResponse到达客户端之前，拦截HttpServletResponse。 根据需要检查HttpServletResponse，也可以修改HttpServletResponse头和数据。

实际上Filter和Servlet极其相似，区别只是Filter不能直接对用户生成响应。实际上Filter里doFilter()方法里的代码就是从多个Servlet的service()方法里抽取的通用代码，通过使用Filter可以实现更好的复用。

Filter和Servlet的生命周期： 1.Filter在web服务器启动时初始化 2.如果某个Servlet配置了 <load-on-startup >1 </load-on-startup >，该Servlet也是在Tomcat（Servlet容器）启动时初始化。 3.如果Servlet没有配置<load-on-startup >1 </load-on-startup >，该Servlet不会在Tomcat启动时初始化，而是在请求到来时初始化。 4.每次请求， Request都会被初始化，响应请求后，请求被销毁。 5.Servlet初始化后，将不会随着请求的结束而注销。 6.关闭Tomcat时，Servlet、Filter依次被注销。

HashMap与HashTable的区别。 1、HashMap是非线程安全的，HashTable是线程安全的。 2、HashMap的键和值都允许有null值存在，而HashTable则不行。 3、因为线程安全的问题，HashMap效率比HashTable的要高。

HashMap的实现机制：

1. 维护一个每个元素是一个链表的数组，而且链表中的每个节点是一个Entry[]键值对的数据结构。
2. 实现了数组+链表的特性，查找快，插入删除也快。
3. 对于每个key,他对应的数组索引下标是 int i = hash(key.hashcode)&(len-1);
4. 每个新加入的节点放在链表首，然后该新加入的节点指向原链表首

HashMap和TreeMap区别

友情链接： Java中HashMap和TreeMap的区别深入理解

HashMap冲突

友情链接： HashMap冲突的解决方法以及原理分析

友情链接： HashMap的工作原理

友情链接： HashMap和Hashtable的区别

友情链接： 2种办法让HashMap线程安全

HashMap，ConcurrentHashMap与LinkedHashMap的区别

1. ConcurrentHashMap是使用了锁分段技术技术来保证线程安全的，锁分段技术：首先将数据分成一段一段的存储，然后给每一段数据配一把锁，当一个线程占用锁访问其中一个段数据的时候，其他段的数据也能被其他线程访问
2. ConcurrentHashMap 是在每个段（segment）中线程安全的
3. LinkedHashMap维护一个双链表，可以将里面的数据按写入的顺序读出

ConcurrentHashMap应用场景

1：ConcurrentHashMap的应用场景是高并发，但是并不能保证线程安全，而同步的HashMap和HashMap的是锁住整个容器，而加锁之后ConcurrentHashMap不需要锁住整个容器，只需要锁住对应的Segment就好了，所以可以保证高并发同步访问，提升了效率。

2：可以多线程写。 ConcurrentHashMap把HashMap分成若干个Segmenet 1.get时，不加锁，先定位到segment然后在找到头结点进行读取操作。而value是volatile变量，所以可以保证在竞争条件时保证读取最新的值，如果读到的value是null，则可能正在修改，那么就调用ReadValueUnderLock函数，加锁保证读到的数据是正确的。 2.Put时会加锁，一律添加到hash链的头部。 3.Remove时也会加锁，由于next是final类型不可改变，所以必须把

删除的节点之前的节点都复制一遍。 4.ConcurrentHashMap允许多个修改操作并发进行，其关键在于使用了锁分离技术。它使用了多个锁来控制对Hash表的不同Segment进行的修改。

ConcurrentHashMap的应用场景是高并发，但是并不能保证线程安全，而同步的HashMap和HashTable的是锁住整个容器，而加锁之后ConcurrentHashMap不需要锁住整个容器，只需要锁住对应的segment就好了，所以可以保证高并发同步访问，提升了效率。

ConcurrentHashMap能够保证每一次调用都是原子操作，但是并不保证多次调用之间也是原子操作。

友情链接：Java集合---ConcurrentHashMap原理分析

Vector和ArrayList的区别

友情链接：Java中Vector和ArrayList的区别

ExecutorService service = Executors.... ExecutorService service = new ThreadPoolExecutor() ExecutorService service = new ScheduledThreadPoolExecutor();

ThreadPoolExecutor源码分析

线程池本身的状态：

```
1.  volatile int runState;
2.  static final int RUNNING = 0;
3.  static final int SHUTDOWN = 1;
4.  static final int STOP = 2;
5.  static final int TERMINATED = 3;
```

等待任务队列和工作集：

```
1.  private final BlockingQueue<Runnable> workQueue; //等待被执行的Runnable任务
2.  private final HashSet<Worker> workers = new HashSet<Worker>(); //正在被执行的Worker任务集
```

线程池的主要状态锁：

```
1.  private final ReentrantLock mainLock = new ReentrantLock();
```

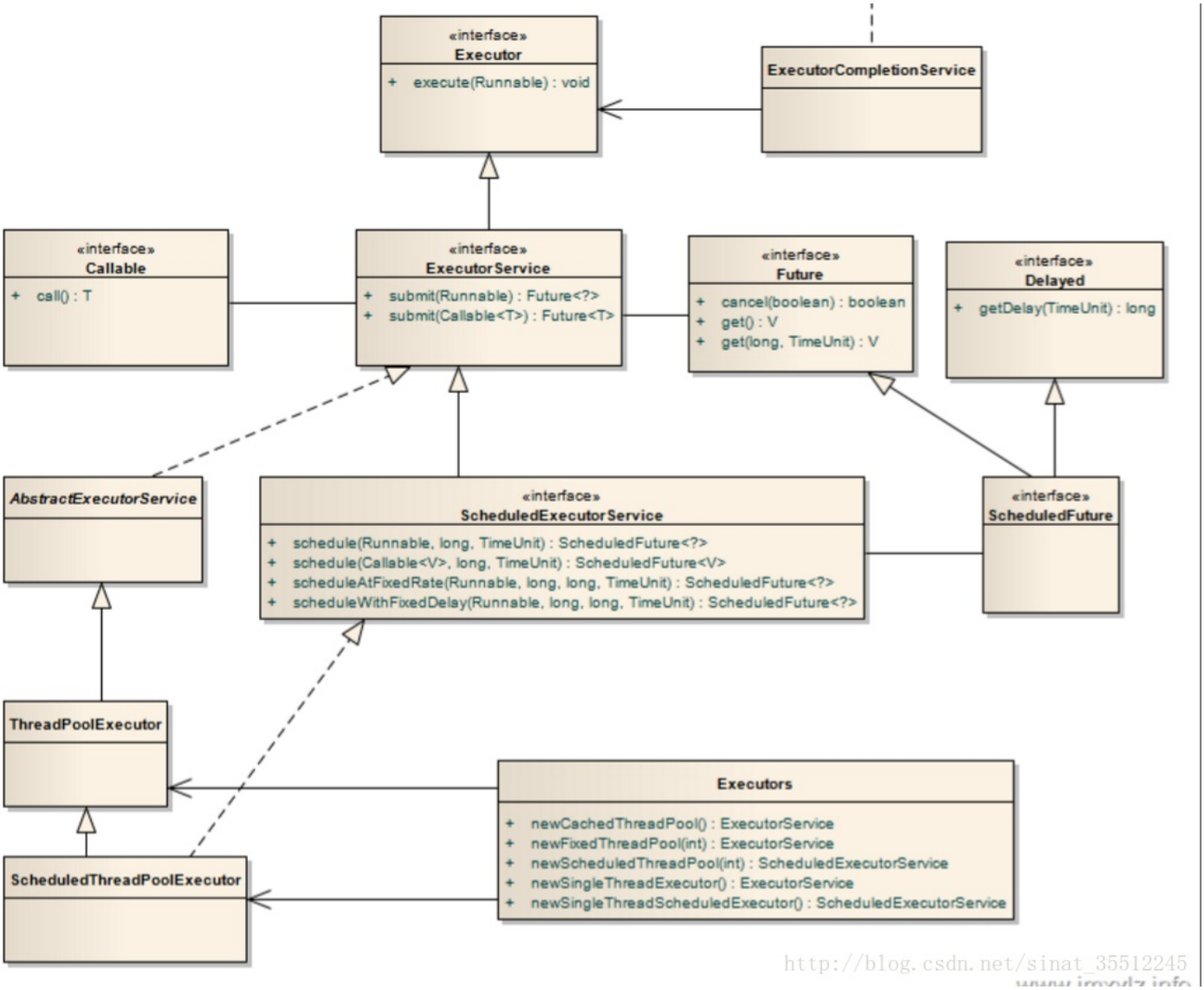
线程池的存活时间和大小：

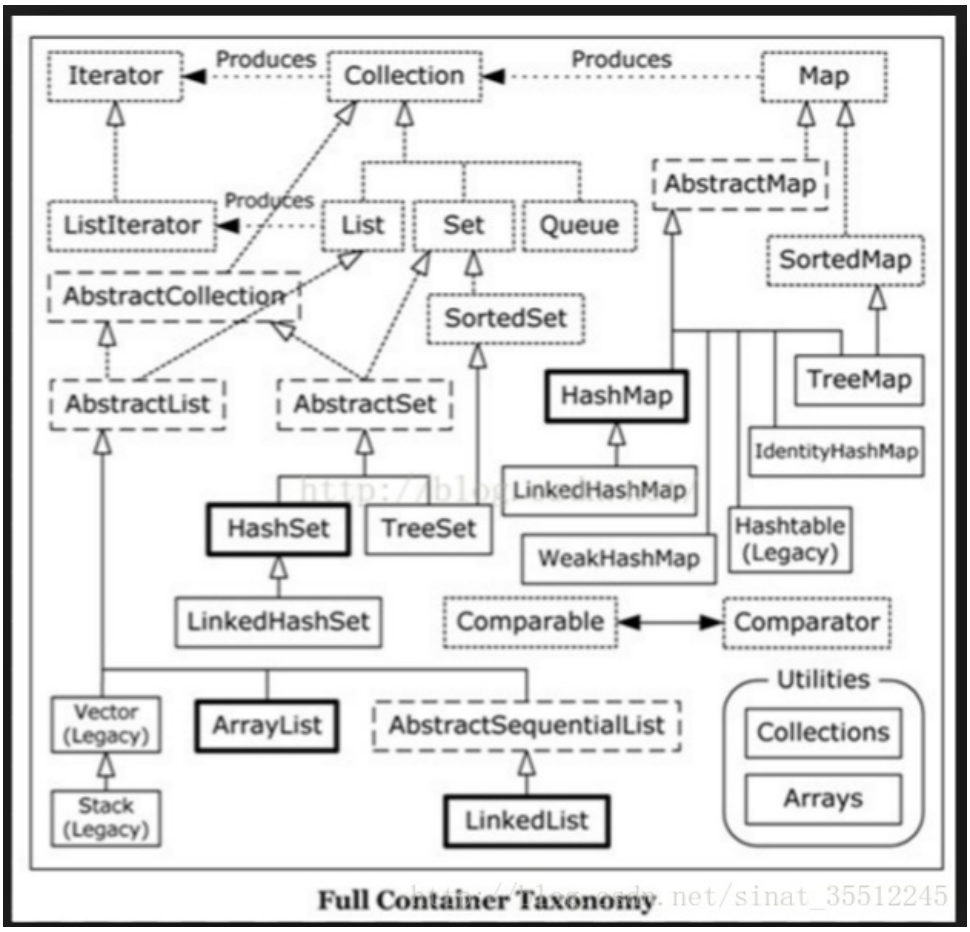
```
1.  private volatile long keepAliveTime;// 线程存活时间
2.  private volatile boolean allowCoreThreadTimeOut;// 是否允许核心线程存活
3.  private volatile int corePoolSize;// 核心池大小
4.  private volatile int maximumPoolSize; // 最大池大小
5.  private volatile int poolSize; //当前池大小
6.  private int largestPoolSize; //最大池大小，区别于maximumPoolSize，是用于记录线程池曾经达到过的最大并发,理论上小于等于maximumPoolSize。
```

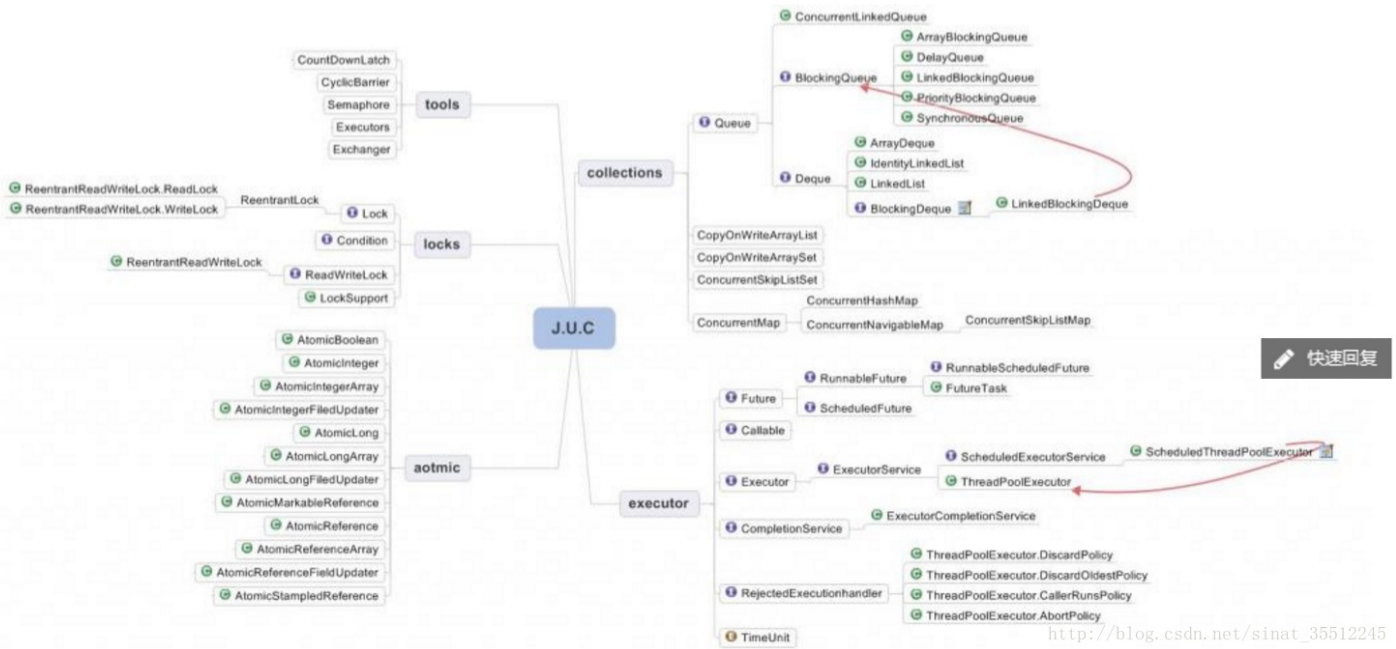
1.2 ThreadPoolExecutor 的内部工作原理 有了以上定义好的数据，下面来看看内部是如何实现的 。 Doug Lea 的整个思路总结起来就是 5 句话：

1. 如果当前池大小 poolSize 小于 corePoolSize ，则创建新线程执行任务。
2. 如果当前池大小 poolSize 大于 corePoolSize ，且等待队列未满，则进入等待队列
3. 如果当前池大小 poolSize 大于 corePoolSize 且小于 maximumPoolSize ，且等待队列已满，则创建新线程执行任务。
4. 如果当前池大小 poolSize 大于 corePoolSize 且大于 maximumPoolSize ，且等待队列已满，则调用拒绝策略来处理该任务。
5. 线程池里的每个线程执行完任务后不会立刻退出，而是会去检查下等待队列里是否还有线程任务需要执行，如果在 keepAliveTime 里等不到新的任务了，那么线程就会退出。

Executor包结构







CopyOnWriteArrayList : 写时加锁，当添加一个元素的时候，将原来的容器进行copy，复制出一个新的容器，然后在新的容器里面写，写完之后再将原容器的引用指向新的容器，而读的时候是读旧容器的数据，所以可以进行并发的读，但这是一种弱一致性的策略。 使用场景：CopyOnWriteArrayList适合使用在读操作远远大于写操作的场景里，比如缓存。

Linux常用命令：cd，cp，mv，rm，ps（进程），tar，cat(查看内容)，chmod，vim，find，ls

死锁的必要条件

1. 互斥 至少有一个资源处于非共享状态
2. 占有并等待
3. 非抢占
4. 循环等待 解决死锁，第一个是死锁预防，就是不让上面的四个条件同时成立。二是，合理分配资源。 三是使用银行家算法，如果该进程请求的资源操作系统剩余量可以满足，那么就分配。

进程间的通信方式

1. 管道(pipe)：管道是一种半双工的通信方式，数据只能单向流动，而且只能在具有亲缘关系的进程间使用。进程的亲缘关系通常是指父子进程关系。
2. 有名管道 (named pipe) ： 有名管道也是半双工的通信方式，但是它允许无亲缘关系进程间的通信。
3. 信号量(semophore) ： 信号量是一个计数器，可以用来控制多个进程对共享资源的访问。它常作为一种锁机制，防止某进程正在访问共享资源时，其他进程也访问该资源。因此，主要作为进程间以及同一进程内不同线程之间的同步手段。
4. 消息队列(message queue) ： 消息队列是由消息的链表，存放在内核中并由消息队列标识符标识。消息队列克服了信号传递信息少、管道只能承载无格式字节流以及缓冲区大小受限等缺点。
5. 信号 (sinal) ： 信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生。
6. 共享内存(shared memory) ：共享内存就是映射一段能被其他进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。共享内存是最快的 IPC 方式，它是针对其他进程间通信方式运行效率低而专门设计的。它往往与其他通信机制，如信号量，配合使用，来实现进程间的同步和通信。
7. 套接字(socket) ： 套解口也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同机器间的进程通信。

进程与线程的区别和联系

操作系统的进程调度算法

计算机系统的层次存储结构详解

数据库事务是指作为单个逻辑工作单元执行的一系列操作。

未提交读（Read uncommitted）	可能	可能	可能
已提交读（Read committed）	不可能	可能	可能
可重复读（Repeatable read）	不可能	不可能	可能
可串行化（Serializable ）	不可能	不可能	不可能

友情链接：数据库事务的四大特性以及事务的隔离级别

MySQL数据库优化总结

MYSQL 优化常用方法

MySQL存储引擎——MyISAM与InnoDB区别

关于SQL数据库中的范式

Hibernate的一级缓存是由Session提供的，因此它只存在于Session的生命周期中，当程序调用save(),update(),saveOrUpdate()等方法 及调用查询接口list,filter,iterate时，如Session缓存中还不存在相应的对象，Hibernate会把该对象加入到一级缓存中，当Session关闭的时候缓存也会消失。
Hibernate的一级缓存是Session所内置的，不能被卸载，也不能进行任何配置一级缓存采用的是key-value的Map方式来实现的，在缓存实体对象时，对象的主关键字ID是Map的key，实体对象就是对应的值。
Hibernate二级缓存：把获得的所有数据对象根据ID放入到第二级缓存中。Hibernate二级缓存策略，是针对于ID查询的缓存策略，删除、更新、增加数据的时候，同时更新缓存。

更新于2017/3/9

Java I/O 总结

JVM（8）：JVM知识点总览-高级Java工程师面试必备

细数JDK里的设计模式

Java中创建对象的5种不同方法

关于Java Collections的几个常见问题

类在什么时候加载和初始化

两个栈实现队列 两个队列实现栈

更新于2017/3/12

java collection.sort()根据时间排序list

单点登录原理与简单实现

更新于2017/3/13

AQS详解

Java的concurrent包

Java 并发工具包 java.util.concurrent 用户指南

更新于2017/6/12

进程和线程的区别：

> 进程：每个进程都有独立的代码和数据空间（进程上下文），进程间的切换会有较大的开销，一个进程包含1--n个线程。
>
> 线程：同一类线程共享代码和数据空间，每个线程有独立的运行栈和程序计数器(PC)，线程切换开销小。
>
> 线程和进程一样分为五个阶段：创建、就绪、运行、阻塞、终止。
>
> 多进程是指操作系统能同时运行多个任务（程序）。
>
> 多线程是指在同一程序中有多个顺序流在执行。

在java中要想实现多线程，有三种手段，一种是继续Thread类，另外一种是实现Runable接口，还有就是实现Callable接口。

Switch能否用string做参数？

a.在 Java 7 之前, switch 只能支持byte,short,char,int 或者其对应的封装类以及 Enum 类型。在Java 7中,String 支持被加上了。

Object有哪些公用方法？

a.方法equals测试的是两个对象是否相等

b.方法clone进行对象拷贝

c.方法getClass返回和当前对象相关的Class对象

d.方法notify,notifyall,wait都是用来对给定对象进行线程同步的

Java的四种引用，强弱软虚，以及用到的场景

a.利用软引用和弱引用解决OOM问题：用一个HashMap来保存图片的路径和相应图片对象关联的软引用之间的映射关系，在内存不足时，JVM会自动回收这些缓存图片对象所占用的空间，从而有效地避免了OOM的问题。

b.通过软可及对象重获方法实现Java对象的高速缓存:比如我们创建了一Employee的类，如果每次需要查询一个雇员的信息。哪怕是几秒中之前刚刚查询过的，都要重新构建一个实例，这是需要消耗很多时间的。我们可以通过软引用和 HashMap 的结合，先是保存引用方面：以软引用的方式对一个Employee对象的实例进行引用并保存该引用到HashMap 上，key 为此雇员的 id，value为这个对象的软引用，另一方面是取出引用，缓存中是否有该Employee实例的软引用，如果有，从软引用中取得。如果没有软引用，或者从软引用中得到的实例是null，重新构建一个实例，并保存对这个新建实例的软引用。

c.强引用：如果一个对象具有强引用，它就不会被垃圾回收器回收。即使当前内存空间不足，JVM也不会回收它，而是抛出 OutOfMemoryError 错误，使程序异常终止。如果想中断强引用和某个对象之间的关联，可以显式地将引用赋值为null，这样一来的话，JVM在合适的时间就会回收该对象。

d.软引用：在使用软引用时，如果内存的空间足够，软引用就能继续被使用，而不会被垃圾回收器回收，只有在内存不足时，软引用才会被垃圾回收器回收。

e.弱引用：具有弱引用的对象拥有的生命周期更短暂。因为当 JVM 进行垃圾回收，一旦发现弱引用对象，无论当前内存空间是否充足，都会将弱引用回收。不过由于垃圾回收器是一个优先级较低的线程，所以并不一定能迅速发现弱引用对象。

f.虚引用：顾名思义，就是形同虚设，如果一个对象仅持有虚引用，那么它相当于没有引用，在任何时候都可能被垃圾回收器回收。

Hashcode的作用，与 equal 有什么区别？

a.同样用于鉴定2个对象是否相等的，java集合中有 list 和 set 两类，其中 set不允许元素重复实现，那个这个不允许重复实现的方法，如果用 equal 去比较的话，如果存在1000个元素，你 new 一个新的元素出来，需要去调用1000次 equal 去逐个和他们比较是否是同一个对象，这样会大大降低效率。hashcode实际上是返回对象的存储地址，如果这个位置上没有元素，就把元素直接存储在上面，如果这个位置上已经存在元素，这个时候才去调用equal方法与新元素进行比较，相同的话就不存了，散列到其他地址上。

Override和Overload的含义以及区别

a.Overload顾名思义是重新加载，它可以表现类的多态性，可以是函数里面可以有相同的函数名但是参数名、返回值、类型不能相同；或者说可以改变参数、类型、返回值但是函数名字依然不变。

b.就是ride(重写)的意思，在子类继承父类的时候子类中可以定义某方法与其父类有相同的名称和参数，当子类在调用这一函数时自动调用子类的方法，而父类相当于被覆盖（重写）了。

具体可前往C++中重载、重写（覆盖）的区别实例分析查看

抽象类和接口的区别

a.一个类只能继承单个类，但是可以实现多个接口

b.抽象类中可以有构造方法，接口中不能有构造方法

c.抽象类中的所有方法并不一定要是抽象的，你可以选择在抽象类中实现一些基本的方法。而接口要求所有的方法都必须是抽象的

d.抽象类中可以包含静态方法，接口中不可以

e.抽象类中可以有普通成员变量，接口中不可以

解析XML的几种方式的原理与特点：DOM、SAX、PULL

a.DOM：消耗内存：先把xml文档都读到内存中，然后再用DOM API来访问树形结构，并获取数据。这个写起来很简单，但是很消耗内存。要是数据过大，手机不够牛逼，可能手机直接死机

b.SAX：解析效率高，占用内存少，基于事件驱动的：更加简单地说就是对文档进行顺序扫描，当扫描到文档(document)开始与结束、元素(element)开始与结束、文档(document)结束等地方时通知事件处理函数，由事件处理函数做相应动作，然后继续同样的扫描，直至文档结束。

c.PULL：与 SAX 类似，也是基于事件驱动，我们可以调用它的next（）方法，来获取下一个解析事件（就是开始文档，结束文档，开始标签，结束标签），当处于某个元素时可以调用XmlPullParser的getAttributte()方法来获取属性的值，也可调用它的nextText()获取本节点的值。

wait()和sleep()的区别

sleep来自Thread类，和wait来自Object类

调用sleep()方法的过程中，线程不会释放对象锁。而 调用 wait 方法线程会释放对象锁

sleep睡眠后不出让系统资源，wait让出系统资源其他线程可以占用CPU

sleep(milliseconds)需要指定一个睡眠时间，时间一到会自动唤醒

JAVA 中堆和栈的区别，说下java 的内存机制

a.基本数据类型比变量和对象的引用都是在栈分配的

b.堆内存用来存放由new创建的对象和数组

c.类变量（static修饰的变量），程序在一加载的时候就在堆中为类变量分配内存，堆中的内存地址存放在栈中

d.实例变量：当你使用java关键字new的时候，系统在堆中开辟并不一定是连续的空间分配给变量，是根据零散的堆内存地址，通过哈希算法换算为一长串数字以表征这个变量在堆中的"物理位置",实例变量的生命周期--当实例变量的引用丢失后，将被GC（垃圾回收器）列入可回收“名单”中，但并不是马上就释放堆中内存

e.局部变量: 由声明在某方法，或某代码段里（比如for循环），执行到它的时候在栈中开辟内存，当局部变量一但脱离作用域，内存立即释放

JAVA多态的实现原理

a.抽象的来讲，多态的意思就是同一消息可以根据发送对象的不同而采用多种不同的行为方式。（发送消息就是函数调用）

b.实现的原理是动态绑定，程序调用的方法在运行期才动态绑定，追溯源码可以发现，JVM 通过参数的自动转型来找到合适的办法。

具体更多资源可前往Java后端面试总结

各种Java面经资源

面试的角度诠释Java工程师（一）

面试的角度诠释Java工程师（二）

Java面试参考指南（一）

Java面试参考指南（二）

阿里面试回来，想和Java程序员谈一谈

面试心得与总结—BAT、网易、蘑菇街

2017年小米春招内推面试面经

历年阿里面试题汇总（2017年不断更新中）

最近5年133个Java面试问题列表

Java的常见误区与细节

Java9都快发布了，Java8的十大新特性你了解多少呢？

欢迎关注个人博客：个人博客

更多内容欢迎关注我的个人公众号

QQ学习交流群（内有干货）：

群名称：程序员聚集地
群　号：677585877