hashcode方法：

在Object类中有一个hashCode方法，native的，hashcode主要用来获取对象的hash值，hashcode方法的主要作用是和散列集合(HashSet，HashMap，HashTable)一起使用

在向集合中插入一个值时(不可重复)，会遍历集合，是否和插入的元素重复，这样效率太低，这时可以使用hash值，当有个对象要插入到集合中时，首先计算出对象的hash值，然后根据一些运算，确定对象的位置，当下一个对象插入时，同样计算出hash值，计算出位置，如果当前位置上有元素，则比较对象是否相等(hash相等，不一定对象相等，对象相等，hash一定相等)，这样就只需比较一次

在重写equals/hashCode方法的时候，一定要验证两个逻辑是否相等

equals和 ==

对于基本数据类型(int，float….)，==比较的就是具体值，而对于引用类型，比较的是其地址是否相等

Object中的equals实际就是==，比较的是地址，而大部分类都对此方法进行重写，首先是通过==比较是否是同一个对象，然后在去判断其元素是否相等

equals和hashcode

hashCode比较的是hash值，而equals用于比较对象的具体值(equals先比较的是地址，再比较元素)

两个对象hashcode相等，equals不一定相等

两个对象equals相等，hashcode可能相等(equals先比较的是地址，再比较元素如果只是元素相等，对象不等)，hashcode就不等了

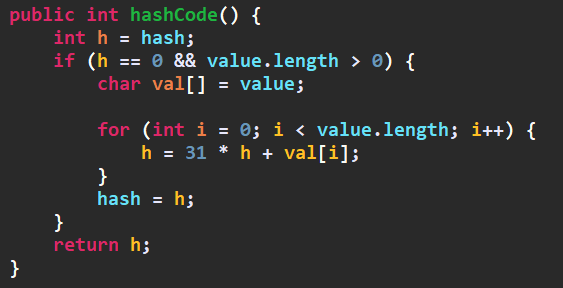
a==b和a.hashcode()==b.hashcode()的区别：

a==b，则a.hashcode()==b.hashcode()恒成立

a!=b，但a.equals(b)=true，则a.hashcode()==b.hashcode()恒成立

这种说法也不一定对，如果类继承的是Object的hashcode，则生成hashcode的策略是根据内存中的地址，如果类重写了hashcode，则有类自己决定，但是大部分生成策略都是根据类中的每个属性生成hash值，比如String的hashcode，是遍历了数组的每个char生成的

如果对于String这类重写了hashcode方法的类来说，上示是成立的，但是没有重写hashcode，使用的是Object默认的hashcode则上示不成立，所以，如果我们在重写hashcode的方法时，应尽量将属性作为hash



a!=b，但a.equals(b)=false，则不确定，可能会出现hash冲突

a.hashcode()==b.hashcode()，无法推出a==b或a.equals(b)=true

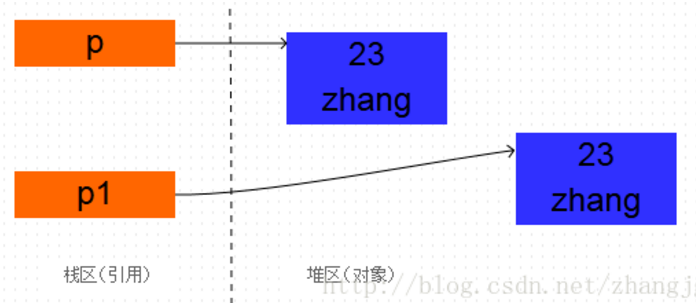
clone

创建对象的方式：new+clone

new是创建一个新的对象，而clone是对以存在的对象进行复制，两个属性完全一样，除了内存中的地址

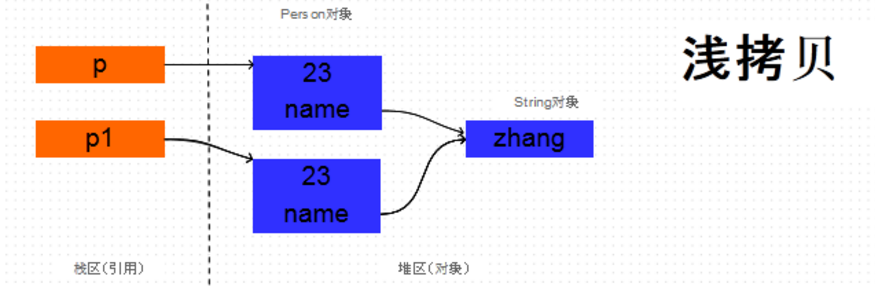
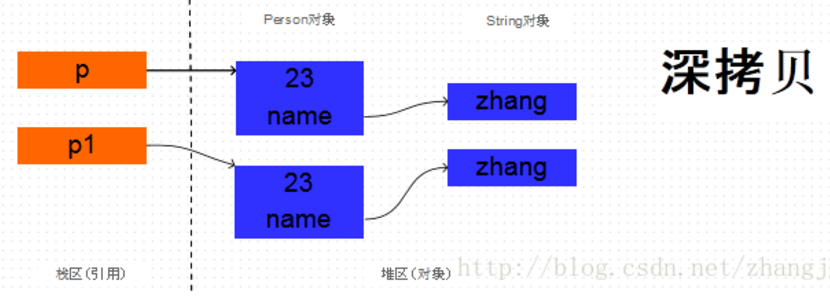
对象能被clone，必须实现Cloneable接口

clone出来的对象不是简单地引用而已，是在内存中真正存在的一个对象

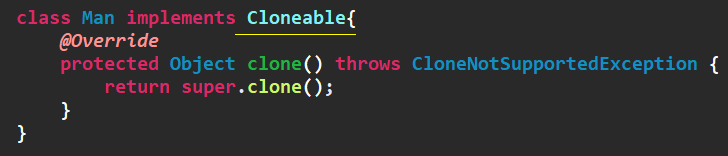
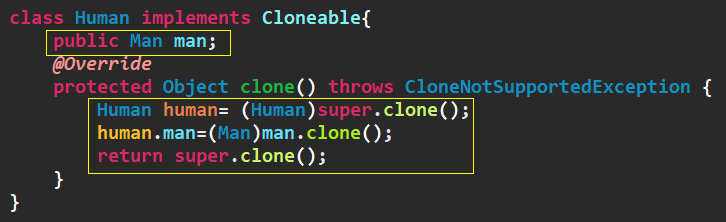


深浅拷贝

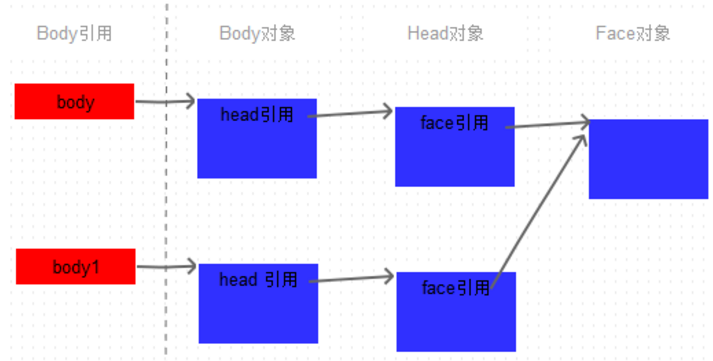
对于基本数据类型，clone执行的深拷贝，而引用类型则是浅拷贝

clone实现深拷贝

这里Human引用Man，在对Human进行clone时，Man是引用类型，所以是浅拷贝，要实现深拷贝就得去clone Man，所以在Human的clone方法中对Man进行了clone，但彻底的深拷贝是不可能实现的，只有是深拷贝，就得一直拷贝下去，直到clone了一个不能clone的类型(比如String)



能被clone，必须实现Cloneable接口，并且重写clone方法，调用父类的clone方法进行clone

String为什么没有clone？没必要，String的字符串本身就是final，常量池中只有一份，clone出来就违背了final的原则

Java常用包：

Java.lang：java进行程序设计的基础包，Object，Math，String，System，Thread，Throwable等

Java.util：集合框架，日期处理

Java.io：文件，流操作

String.split()只有String采用，StringBuffer没有

String str=" We Are Happy "

str.split(" "):分割到Happy后就不会往后分割，所以分割的结果为"","","We","Are","Happy"

str.split(" ",n)

n表示分割出来串的个数

n=0:表示不会分割后面的，即结果为"","","We","Are","Happy"

n=1：表示分割出来的串只有一个，即原来的串" We Are Happy "

n=2：2个串，"" " We Are Happy "

依次推，分割出来串最多的个数为str.lengt()-1

所以要想分割happy后的"",可以声明n=str.lengt()-1

Comparator和Comparable接口之间区别？

Comparable接口中只有一个public int compareTo(T o)方法，用于比较两个对象的大小

Comparator接口中有很多static方法(jdk1.8以后，接口中可以有静态方法的实现体)和boolean equals(Object obj)/

int compare(T o1, T o2)两个抽象方法

compareTo更适用于String，基本类型包装类中比较两个数的大小

Comparator定义的一个比较器，重点是器，一个模板，常在Collection集合中对集合元素进行排序

Hash的负载因子为什么是0.75？

一方面是大量实验的结果，还有就是0.75\*capacity(2的幂)的结果为整数

cookie的内容主要包括：名字，值，过期时间，路径和域。路径与域一起构成cookie的作用范围。

Cookie中只能保管ASCII字符串，假如需求存取Unicode字符或者二进制数据

Session中能够存取任何类型的数据，运用起来十分便当。能够把Session看做是一个Java容器类。

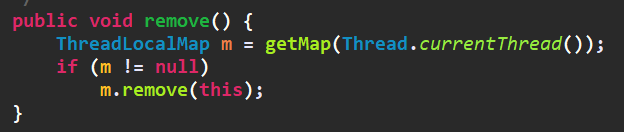
Cookie支持跨域名访问，例如将domain属性设置为“.biaodianfu.com”，则以“.biaodianfu.com”为后缀的一切域名均能够访问该Cookie。

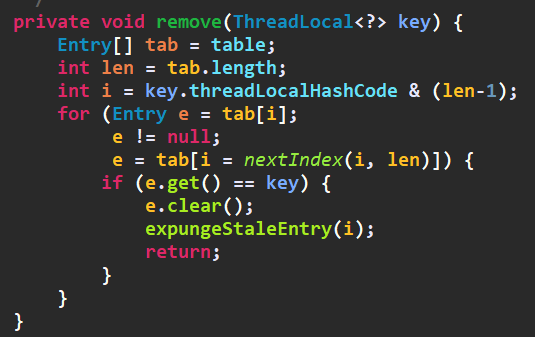
ThreadLocal提供的只是一个浅拷贝，可以通过重写ThreadLocal的initialValue()函数来自己实现深拷贝

Spring与线程安全：

对于作用域为Singleton的Bean而言是无状态的是不存在线程安全问题，对于某些bean需要线程共享的bean可以存放在ThreadLocal

ThreadLocal内存泄漏，如果ThreadLocal置为null，会导致Thread中的ThreadLocalMap的key就为null了，但是value值任然存在，而不会被引用，导致一直占有内存，所以在将ThreadLocal置为null之前，可以调用ThreadLocal的remove方法，将对应ThreadLocal的value从ThreadLocalMap中移出(ThreadLocalMap是一个数)





Singleton的Bean在Spring中的生命周期：

实例化bean

引入依赖(IOC)

如果Bean实现BeanNameAware接口，会调用setBeanName(String beanId)，将Spring配置文件中的Bean的ID传递给Bean

如果Bean实现BeanFactoryAware接口，会调用setBeanFactory()，传递的是Spring工厂本身

如果Bean实现ApplicationContextAware接口，会调用setApplicationContext(ApplicationContext)方法，传递的是ApplicationContext对象，用于获取其他Bean(和BeanFactory类似)

调用实现BeanPostProcessor接口的Bean(Bean后置处理器)的postProcessBeforeInitialization(Object String)方法

如果Bean实现InitializingBean接口，调用afterPropertiesSet对bean操作

如果配置文件中配置了init-method属性，调用指定的方法处理bean(和InitializingBean接口一样)

调用实现BeanPostProcessor接口的Bean(Bean的后置处理器)的postAfterInitialization(Object，String)方法

Bean随便使用

当Bean被销毁之前

如果bean实现DisposableBean接口，会调用destroy方法

如果配置文件中声明destroy-method属性

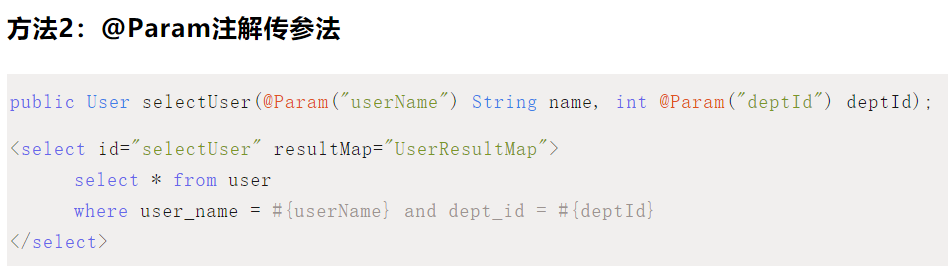
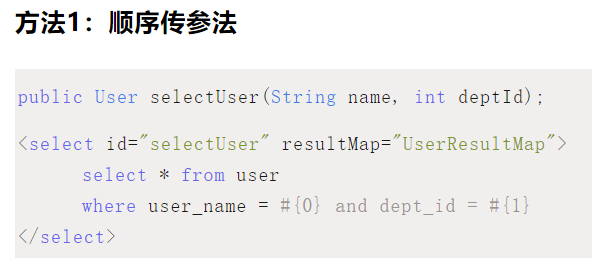
Mybatis缓存：

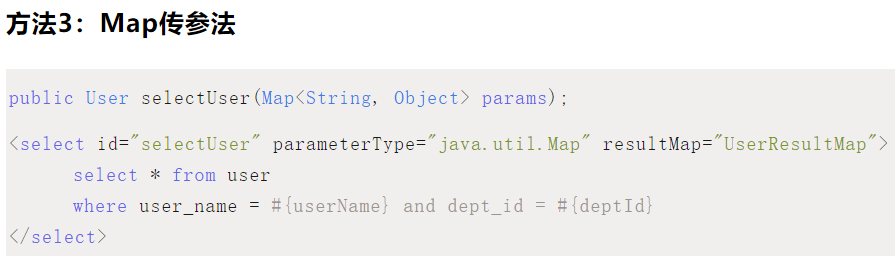
一级缓存的生命周期和SqlSession一致(多个SqlSession之间会导致幻读，重复读)

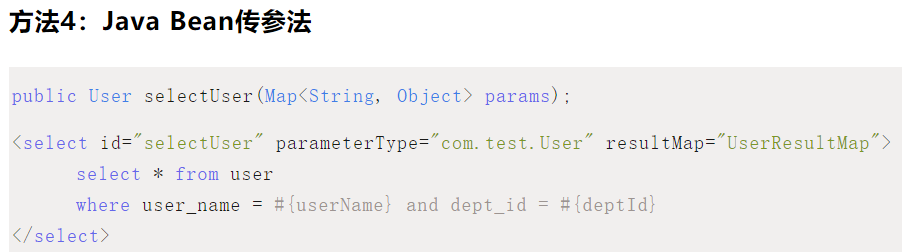
一级缓存内部是通过HashMap实现

二级缓存实现了SqlSession之间缓存数据的共享

Mybatis传递参数的方法：







可以在Spring中注入一个null 和一个空字符串

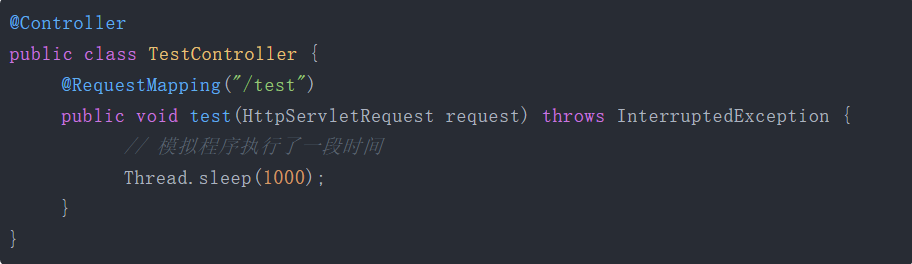
在客户端，一个浏览器能创建的 Cookie 数量最多为 300 个，并且每个不能超过 4KB，每个 Web 站点能设置的 Cookie 总数不能超过 20 个

事务：

JDBC事务主要优点是API简单，性能较好，但是JDBC事务不支持跨数据，所以分布式场景不能使用JDBC

JTA事务支持分布式事务，跨域访问，但是JTA很笨重

在Controller获取Request的方法以及线程安全：



线程安全，这里的request是方法的局部变量，每个线程唯一拥有，只有多个线程对类的属性进行操作是才会出现线程安全问题



线程安全，Request看似是类的属性，但是其实这里实际是一个Singleton的bean调用了一个Property的bean，所以每次执行方法的时候，获取的都是新的request

为什么我们调用start()方法时会执行run()方法，为什么我们不能直接调用run()方法？

当调用start方法时，才是创建线程，线程会自动执行run方法，而New Thread只是声明一个线程