**Servlet生命周期和工作原理**

**Servlet生命周期分为三个阶段：**

1：初始化阶段，调用init()方法

2：响应客户请求阶段，调用service()方法

3：终止阶段，调用destory()方法

**工作原理**

先解释Servlet接收和响应客户请求的过程，首先客户发送个请求，Servlet容器会创建特定于这个请求的ServletRequest对象和ServletResponse对象，然后调用Servlet的service()方法。Service()方法从ServletRequest对象获得客户请求信息，处理该请求，然后通过ServletResponse对象向客户返回响应信息

**GC详解**

GC，即就是Java垃圾回收机制。目前主流的JVM（HotSpot）采用的是分代收集算法。与C++不同的是，Java采用的是类似于树形结构的**可达性分析法**来判断对象是否还存在引用。即：从gcroot开始，把所有可以搜索得到的对象标记为存活对象。

GC机制

要准确理解Java的垃圾回收机制，就要从：“什么时候”，“对什么东西”，“做了什么”三个方面来具体分析。

第一：“什么时候”即就是GC触发的条件。GC触发的条件有两种。（1）程序调用System.gc时可以触发(会建议JVM进行垃圾回收，不代表一定会进行GC)；（2）系统自身来决定GC触发的时机。 系统判断GC触发的依据：根据Eden区和From Space区的内存大小来决定。当内存大小不足时，则会启动GC线程并停止应用线程。

第二：“对什么东西”笼统的认为是Java对象并没有错。但是准确来讲，GC操作的对象分为：通过可达性分析法无法搜索到的对象和可以搜索到的对象。对于搜索不到的方法进行标记。

第三：“做了什么”最浅显的理解为释放对象。但是从GC的底层机制可以看出，对于可以搜索到的对象进行复制操作，对于搜索不到的对象，调用finalize()方法进行释放。 具体过程：当GC线程启动时，会通过可达性分析法把Eden区和From Space区的存活对象复制到To Space区，然后把Eden Space和From Space区的对象释放掉。当GC轮训扫描To Space区一定次数后，把依然存活的对象复制到老年代，然后释放To Space区的对象。

JVM加载Class文件的原理和机制

JVM中的类的加载是由ClassLoader（类加载器）和它的子类来实现的，

Java CLassloader是一个很重要的java运行时的系统组件，它负责在运行时查找和装入类文件的类。java中的所有类，都需要装载到JVM中才能运行。

类加载器本身就是一个类，他的工作就是将class文件从硬盘读取到内存中。

Cookie 和 Session的区别

共同之处：cookie和session都是用来跟踪浏览器用户身份的会话技术

cooKie：客户端会话技术，将数据保存在客户端。

session：服务器端会话技术，在一次会话的多次请求间共享数据，将数据保存在服务器端的对象中。

但是Session的实现依赖于Cookie。

session与Cookie的区别：

1. session存储数据在服务器端，Cookie在客户端

2. session没有数据大小限制，Cookie

有单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

3. session数据安全，Cookie相对于不安全

forward(转发)和redirect(重定向)的区别

forward转发：

1. 转发的地址不发生改变
2. 转发的请求是一次请求，可使用request对象来共享数据
3. 只能访问当前服务器下的资源文件

redirect 重定向：

1. 地址发生改变
2. 重定向的请求是二次请求，不可以使用reques对象来共享数据
3. 重定向可以访问其他服务器下的资源文件。

集合

ArrayList

底层数据结构是数组，查询快，增删慢，线程不安全，效率高

（数组查询具有所有查询特定元素比较快。而插入和删除和修改比较慢(数组在内存中是一块连续的内存，如果插入或删除是需要移动内存）

LinkedList

底层数据结构是链表，查询慢，增删块，线程安全，效率低

（链表不要求内存是连续的，在当前元素中存放下一个或上一个元素的地址。查询时需要从头部开始，一个一个的找。所以查询效率低。插入时不需要移动内存，只需改变引用指向即可。所以插入或者删除的效率高。）

Vector

底层数据结构是数组，查询快，增删慢，线程不安全，效率高