# 计算机网络

## OSI七层协议：

应用层：为应用程序提供网络服务；

表示层：数据格式转换、数据压缩和数据加密；

会话层：建立、断开和维护通信链接；

传输层：为上层协议提供端到端的可靠传输；

网络层：寻址和路由；

数据链路层：定义通过通信媒介互连的设备之间传输的规范；

物理层：利用物理传输介质为数据链路层提供物理连接。；

## TCP五层模型：

将OSI的应用层，表示层，会话层合并为应用层；其他不变

## TCP/IP四层协议：

在TCP五层协议基础上将物理层和数据链路层合并为网络接口层

## TCP协议：

transmission control protocol 负责应用软件（比如你的浏览器）和网络软件之间的通信。负责将数据分割并装入 IP 包，然后在它们到达的时候重新组合它们。采用字节流传输

TCP流量控制采用滑动窗口机制，即流量控制就是让发送方的发送速率不要太快，要让接收方来得及接收。

TCP拥塞控制采用慢开始和拥塞避免；快重传和快恢复。

## IP协议：

Internet protocol IP协议包含了IP地址，IP协议只能发送无连接，不可靠的数据包

## IP地址

IP协议分为IPv4和IPv6版本，**IPv4的IP地址由4字节32比特构成；分为了网络号（包括子网号）和主机号**；网络地址是全网唯一的，用于标定在互联网中的位置。

### 网络号

网络号就是我们常说的A类地址、B类地址、C类地址、D类地址、E类地址，由IP地址中的前数位构成。

A类地址：8位网络号(0开头）+24位主机号，范围：1.0.0.0 - 127.255.255.255

B类地址：16位网络号（10开头）+16位主机号，范围：128.0.0.0 - 191.255.255.255

C类地址：24位网络号（110开头）+8位主机/号，范围：192.0.0.0 - 223.255.255.255

D类：组播地址，范围：224.0.0.0 - 239.255.255.255

E类：保留以后再用，范围：240.0.0.0 - 255.255.255.255

**在以上点分十进制的IP地址段中没有127.0.0.0/8这个网段，因为该网段被用作环回地址。**

**主机号全为0代表该网络的网络号，主机号全为1代表该网络的广播地址，都不能分配给主机。**

A类地址所能容纳的主机数最多，到了后期为了解决网络号占位太多，主机号不足的问题，对一个高类别的IP地址进行再划分，形成多个子网。

### 子网掩码

子网掩码是用于标志2个IP地址是否属于同一子网的，也是32位二进制地址，其网络位都为1，主机位都为0；用于与IP地址与运算，得到网络号。

## MAC地址

MAC地址（Media Access Control Address）的全称叫做媒体访问控制地址，也称作局域网地址，以太网地址或者物理地址。

MAC地址应用在OSI模型的数据链路层，**是每一个网卡独有的地址。**

## ARP协议

地址解析协议，即ARP（Address Resolution Protocol），是根据IP地址获取物理地址的一个TCP/IP协议。

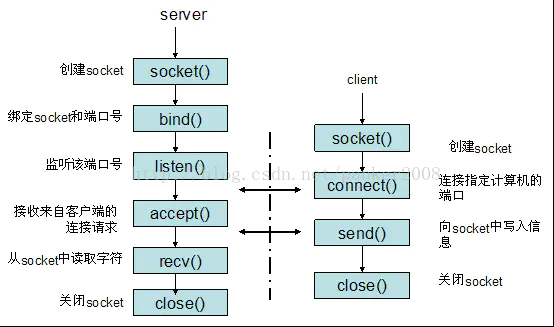
## Socket

socket 被翻译为“套接字”，它是计算机之间进行通信的一种约定或一种方式。通过 socket 这种约定，一台计算机可以接收其他计算机的数据，也可以向其他计算机发送数据一种特殊文件用来标识。即连接建立、数据传输等操作都是通过该Socket()函数操作的一些socket文件实现的。

Socket通信的数据传输方式，常用的有两种：

　　ａ、SOCK\_STREAM：表示面向连接的数据传输方式。数据可以准确无误地到达另一台计算机，如果损坏或丢失，可以重新发送，但效率相对较慢。常见的 http 协议就使用 SOCK\_STREAM 传输数据，因为要确保数据的正确性，否则网页不能正常解析。

　　ｂ、SOCK\_DGRAM：表示无连接的数据传输方式。计算机只管传输数据，不作数据校验，如果数据在传输中损坏，或者没有到达另一台计算机，是没有办法补救的。也就是说，数据错了就错了，无法重传。因为 SOCK\_DGRAM 所做的校验工作少，所以效率比 SOCK\_STREAM 高。



### 断开连接

close()/closesocket()和shutdown()的区别

close() / closesocket() 用来关闭套接字，将套接字描述符（或句柄）从内存清除，应用程序关闭套接字后，与该套接字相关的连接和缓存也失去了意义。

shutdown() 用来关闭连接，而不是套接字，不管调用多少次 shutdown()，套接字依然存在，直到调用 close() / closesocket() 将套接字从内存清除。

默认情况下，close()/closesocket() 会立即向网络中发送FIN包，不管输出缓冲区中是否还有数据，而shutdown() 会等输出缓冲区中的数据传输完毕再发送FIN包。也就意味着，调用 close()/closesocket() 将丢失输出缓冲区中的数据，而调用 shutdown() 不会

## Cookie

HTTP协议本身是无状态的。什么是无状态呢，即服务器无法判断用户身份。Cookie实际上是一小段的文本信息（key-value格式）。客户端向服务器发起请求，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务器。服务器检查该Cookie，以此来辨认用户状态。

### cookie机制

当用户第一次访问并登陆一个网站的时候，cookie的设置以及发送会经历以下4个步骤：

客户端发送一个请求到服务器 --》 服务器发送一个HttpResponse响应到客户端，其中包含Set-Cookie的头部 --》 客户端保存cookie，之后向服务器发送请求时，HttpRequest请求中会包含一个Cookie的头部 --》服务器返回响应数据

### Cookie属性

NAME=VALUE 键值对，可以设置要保存的 Key/Value，注意这里的 NAME 不能和其他属性项的名字一样

Expires 过期时间，在设置的某个时间点后该 Cookie 就会失效

Domain 生成该 Cookie 的域名，如 domain="www.baidu.com"

Path 该Cookie 是在当前的哪个路径下生成的，如 path=/wp-admin/

Secure 如果设置了这个属性，那么只会在 SSH 连接时才会回传该 Cookie

Cookie中的maxAge用来表示Cookie的有效期, 单位为秒。Cookie中通过getMaxAge()和setMaxAge(int maxAge)来读写该属性。maxAge有3种值，分别为正数，负数和0。

如果maxAge属性为正数，则表示该Cookie会在maxAge秒之后自动失效。

当maxAge属性为负数，则表示该Cookie只是一个临时Cookie，关闭浏览器后立即失效、

当maxAge为0时，表示立即删除Cookie

maxAge只是一个只读属性

### 修改或者删除Cookie

HttpServletResponse提供的Cookie操作只有一个addCookie(Cookie cookie)，所以想要修改Cookie只能使用一个同名的Cookie来覆盖原先的Cookie。如果要删除某个Cookie，则只需要新建一个同名的Cookie，并将maxAge设置为0，并覆盖原来的Cookie即可。



注：**seq**:sequence,序列号；**ack**:acknowledge，确认号；**syn**：synchronize，请求同步标志；

**fin**：finally，结束标志。

## 三次握手：

**也称连接建立阶段。**client端发送连接请求报文syn，server端收到后回复ack确认并分配资源+syn请求连接，client端收到后再回复ack确认并分配资源。

**具体过程：client端发送SYN=1,seq=x(随机选择)到server端，并进入SYN\_SENT态。server端收到后进入SYN\_RCVD态，回复SYN=1,ACK=1,seq=y(随机选择)，ack=x+1。client端收到回复后发送ACK=1,ack=y+1并进入ESTABLISHED态，server端收到后也进入ESTABLISHED态。**

## SYN攻击

在三次握手过程中，server端发送SYN+ACK后到收到client端回复的ACK之前这段时间称为半连接，此时server端进入Syn-RECV态，当收到ACK后进入ESTABLISHED态。

SYN攻击就是短时间伪造大量不存在的IP地址，不停向server端发送SYN包，server端收到后回复SYN+ACK包，并等待回复ACK包，但由于源地址不存在，服务器将一直重发,此时伪造的SYN包将长时间占用server端的未连接队列，正常的SYN包请求将被排挤，目标server端运行缓慢，严重可引起网络堵塞乃至系统瘫痪

检测SYN攻击的方式非常简单，即当Server上有大量半连接状态且源IP地址是随机的，则可以断定遭到SYN攻击了

## 四次握手：

也称连接释放阶段。client发起中断连接请求，即发送fin报文；server端收到后先回复ack，由于此刻server端可能还在发送data，所以之后才回复fin报文，client端收到后发送ack报文并进入TIME\_WAIT状态。server端收到ack后就可以断开连接了。此时当client端等待2MSL后未收到回复则默认server端已关闭，此时client端也可以关闭连接了。

### 为什么会进入TIME\_WAIT状态，不直接结束？

1. 若最后一次发送端的ACK包丢失，则接收端会重新发起FIN包，而此时发送端已关闭，则接收端永远接受不到ACK包。最后收到RST,不符合TCP协议的可靠连接要求
2. 若关闭连接后又马上发起一个新连接，当新连接与旧连接端口相同时

### MSL（Maximum Segment Lifetime）

MSL即报文最大生存时间，**数据包在网络中是有生存时间的，**超过这个时间还未到达目标主机就会被丢弃，并通知源主机。**2MSL 是数据包往返的最大时间**

## 为什么要三次握手不能两次握手：

1. 可能发生死锁，即server端的ACK包丢失后client将一直等待server端的ACK包，忽略任何server端发过来的数据包。
2. 有时候client端第一次发送SYN时可能会丢失，所以长时间（实际很短）未收到回复则会再发送一次SYN。此时server端先后收到2次SYN请求，会建立两个连接，但实际上client端只用了一个，极端情况下会有N多连接得不到响应，极大浪费资源。

## 为什么建立连接协议是三次握手，关闭连接是四次握手？

## DNS协议

**DomainNameSystem,域名系统缩写**。将url转换为IP地址

## 什么是UDP和TCP？UDP和TCP的区别是什么？

UDP和TCP都是传输层的协议，用来建立可靠的通信传输链接的。

UDP仅提供了最基本的数据传输功能，至于传输时连接的建立和断开、传输可靠性的保证这些UDP统统不关心，而是把这些问题抛给了UDP上层的应用层程序去处理，自己仅提供传输层协议的最基本功能。

**TCP作为一种面向有连接的协议，只有在确认通信对端存在时才会发送数据**，会在传输开始前建立连接，传输结束后断开连接，此外，TCP还采取了多种措施保障传输的可靠性。

**TCP和UDP的区别如下：**

TCP是面向有连接型，UDP是面向无连接型；

TCP是一对一传输，UDP支持一对一、一对多、多对一和多对多的交互通信；

TCP是面向字节流的，即把应用层传来的报文看成字节流，将字节流拆分成大小不等的数据块，并添加TCP首部；UDP是面向报文的，对应用层传下来的报文不拆分也不合并，仅添加UDP首部；

TCP支持传输可靠性的多种措施，包括保证包的传输顺序、重发机制、流量控制和拥塞控制；UDP仅提供最基本的数据传输能力。

## TCP对应的应用层协议有哪些？UDP对应的应用层协议有哪些？

TCP对应的典型的应用层协议：

FTP：文件传输协议；

SSH：远程登录协议；

HTTP：web服务器传输超文本到本地浏览器的超文本传输协议。

UDP对应的典型的应用层协议：

DNS：域名解析协议；

TFTP：简单文件传输协议；

SNMP：简单网络管理协议。

## 一个页面从输入url到加载完成的过程都发生了什么，请详细说明

1、首先，在浏览器地址栏中输入url

2、浏览器先查看浏览器缓存-系统缓存-路由器缓存，如果缓存中有，会直接在屏幕中显示页面内容。若没有，则跳到第三步操作

3、在发送http请求前，需要域名解析(DNS解析)，解析获取相应的IP地址

4、浏览器向服务器发起tcp连接，与浏览器建立tcp三次握手

5、握手成功后，浏览器向服务器发送http请求，请求数据包

6、服务器处理收到的请求，将数据返回至浏览器

7、浏览器收到HTTP响应

8、浏览器解析渲染页面:解析html源码;生成Dom树、解析css样式、js交互

# HTTP协议

### 协议概念

需掌握前置技能servlet

**HTTP 超文本传输协议 Hyper Text Transfer Protocol**  
  
当我们在浏览器的地址栏输入一个地址的时候，就能够访问服务器的某个页面  
这个过程本身就是两个应用程序之间的交互，一个应用程序是浏览器，另一个应用程序是服务器  
协议是什么？ 协议就是不同的应用程序之间按照事先做好的约定进行的通信。 这样就能互相读懂对方的意思。  
浏览器和WEB服务器之间，使用的就是一种叫做HTTP的协议。 这样是BS (Browser Server )架构模型的基础

**HTTP协议由请求和响应两种类型组成**

请求消息 由 请求行，请求头部，空行和请求数据4部分组成

响应信息由状态行，消息报头，响应正文组成

http协议中的响应代码从 1xx ~ 5xx，一共有41种，不过很多状态码很少会接触到。

200 表示响应成功了，比如打开页面, 就会看到200响应码

301 表示客户端跳转，永久性的跳转  
为了实现301跳转，在Servlet中应该使用如下代码：

response.setStatus(301);

response.setHeader("Location", "fail.html");

# HTTP

**超文本传输协议（英文：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。**HTTP是万维网的数据通信的基础。

HTTP是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。。通过使用网页浏览器、网络爬虫或者其它的工具，**客户端发起一个HTTP请求到服务器上指定端口（默认端口为80）**。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如HTML文件和图像。我们称这个应答服务器为源服务器（origin server）。

HTTP可以在任何互联网协议上，或其他网络上实现。HTTP假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。

## HTTP请求方法

HTTP/1.1协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源：

**GET**

向指定的资源发出“显示”请求。使用GET方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在Web Application中。其中一个原因是GET可能会被网络蜘蛛等随意访问。

**HEAD**

与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

**POST**

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。

**PUT**

向指定资源位置上传其最新内容。

**DELETE**

请求服务器删除Request-URI所标识的资源。

**TRACE**

回显服务器收到的请求，主要用于测试或诊断。

**OPTIONS**

这个方法可使服务器传回该资源所支持的所有HTTP请求方法。用'\*'来代替资源名称，向Web服务器发送OPTIONS请求，可以测试服务器功能是否正常运作。

**CONNECT**

HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。通常用于SSL加密服务器的链接（经由非加密的HTTP代理服务器）。

### GET和POST区别

1、GET请求一般用去请求获取数据，

POST一般作为发送数据到后台时使用

2、GET产生一个TCP数据包，将header和data合并成一次请求完成；POST产生两个TCP数据包。分别传送header和data。

## 请求头部

Host: 主机名

User-Agent: 浏览器基本资料

Accept: 浏览器能够识别的响应类型

Accept-Language: 浏览器默认语言

Accept-Encoding: 浏览器能够识别的压缩方式

Referer: 来路页面， /addHero 这个路径是通过addHero.html这个页面跳转过来的。

Connecton：是否保持连接

## 消息报头

消息报头中提供如下信息：

Content-Length: 表示长度

Content-Type: 内容格式

Date: 日期

Server: 服务器类型

## 常见的http返回码有哪些？

**1\*\* 信息，服务器收到请求，需要请求者继续执行操作**

**2\*\* 成功，操作被成功接收并处理**

**3\*\* 重定向，需要进一步的操作以完成请求**

**4\*\* 客户端错误，请求包含语法错误或无法完成请求**

**5\*\* 服务器错误，服务器在处理请求的过程中发生了错误**

**200：请求被正常处理**

**204：请求被受理但没有资源可以返回**

206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。

**301：永久性重定向**

**302：临时重定向**

303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上

**304：Not Modified 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。**

307：临时重定向，与302类似，只是强制要求使用POST方法

400：请求报文语法有误，服务器无法识别

401：请求需要认证

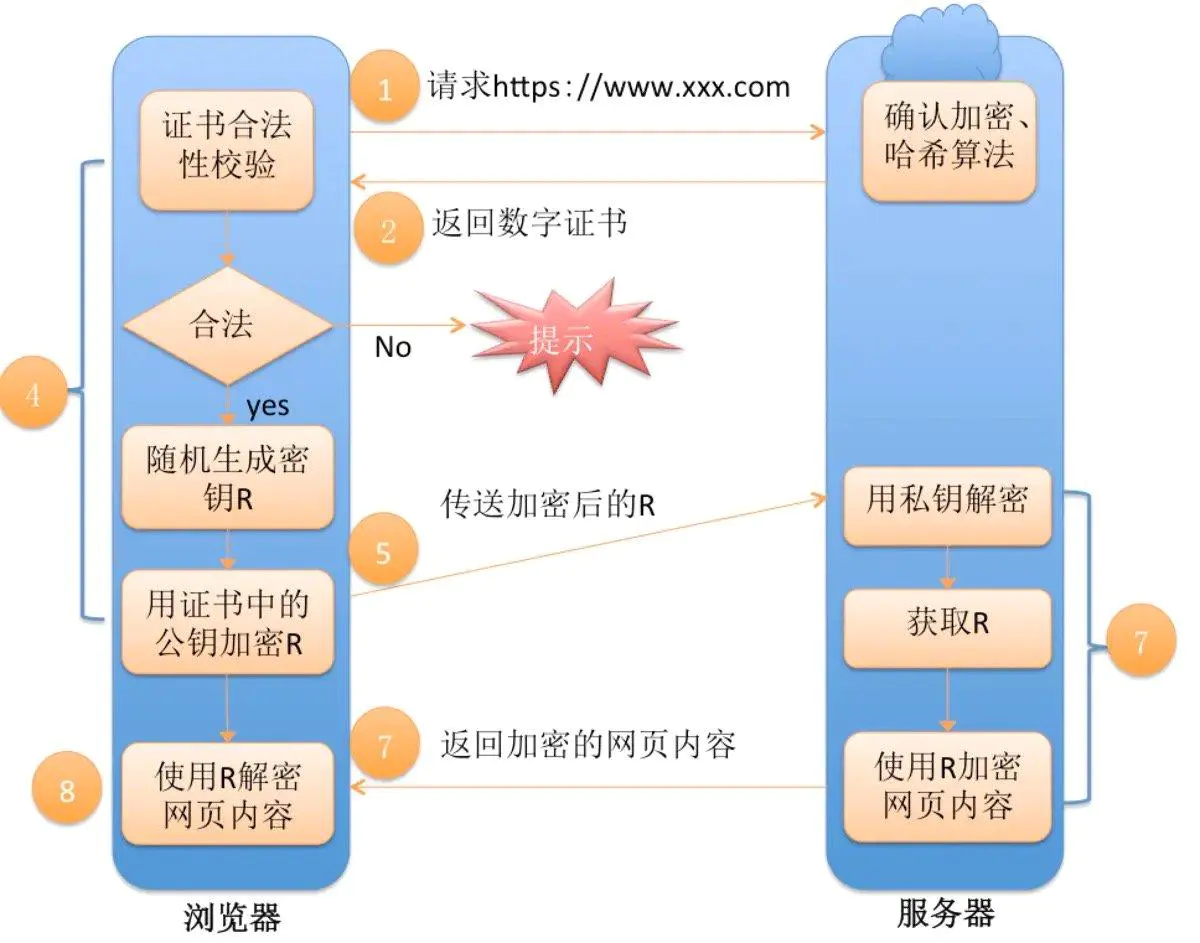
**403：请求的对应资源禁止被访问**

**404：服务器无法找到对应资源**

**500：服务器内部错误**

503：服务器正忙

## HTTPS请求过程



HTTPS实际上就是HTTP over SSL，它使用默认端口443。HTTPS协议使用SSL在发送方把原始数据进行加密，然后在接受方进行解密，加密和解密需要发送方和接受方通过交换共知的密钥来实现，因此，所传送的数据不容易被网络黑客截获和解密。

**具体过程：在三次握手建立连接后，client端发送请求到server端，server端返回数字证书给client端，client端确认数字证书有效后随机生成属于自己的秘钥R，并用证书中的公钥加密，发送给server端，server端用证书的私钥解密得到本次密钥R，并用R加密内容后发送给client端，client端再用R解密。**

### SSL加密

（Secure Socket Layer，安全套接字层）位于TCP/IP协议与各种应用层协议之间，为数据通讯提供安全支持。分为SSL记录协议和SSL握手协议。前者为高层协议提供数据封装、压缩、**加密**等基本功能的支持。后者用于在实际的数据传输开始前，通讯双方进行**身份认证、**协商加密算法、交换加密密钥等。SSL握手需要额外的9个包，证书其实就是由一套私钥和公钥构成。

## 端口

80

443

## http2.0的特性如下：

1、内容安全，应为http2.0是基于https的，天然具有安全特性，通过http2.0的特性可以避免单纯使用https的性能下降

2、二进制格式，http1.X的解析是基于文本的，http2.0将所有的传输信息分割为更小的消息和帧，并对他们采用二进制格式编码，基于二进制可以让协议有更多的扩展性，比如引入了帧来传输数据和指令

3、多路复用，这个功能相当于是长连接的增强，每个request请求可以随机的混杂在一起，接收方可以根据request的id将request再归属到各自不同的服务端请求里面，另外多路复用中也支持了流的优先级，允许客户端告诉服务器那些内容是更优先级的资源，可以优先传输，

# JVM

## JVM定义

JVM是Java Virtual Machine的缩写。它是一种基于计算设备的规范，是一台虚拟机，即虚构的计算机。

JVM屏蔽了具体操作系统平台的信息（显然，就像是我们在电脑上开了个虚拟机一样），当然，**JVM执行字节码时实际上还是要解释成具体操作平台的机器指令的。**

通过JVM，Java实现了平台无关性，Java语言在不同平台运行时不需要重新编译，只需要在该平台上部署JVM就可以了。因而能实现一次编译多处运行。(就像是你的虚拟机也可以在任何安了VMWare的系统上运行)

## JVM的GC回收机制

将内存中不再被使用的对象进行回收，GC中用于回收的方法称为收集器，由于GC需要消耗一些资源和时间，Java在对对象的生命周期特征进行分析后，按照新生代、旧生代的方式来对对象进行收集，以尽可能的缩短GC对应用造成的暂停。

不同的对象引用类型， GC会采用不同的方法进行回收，JVM对象的引用分为了四种类型：

强引用：默认情况下，对象采用的均为强引用（这个对象的实例没有其他对象引用，GC时才会被回收）。

软引用：软引用是Java中提供的一种比较适合于缓存场景的应用（只有在内存不够用的情况下才会被GC）。

弱引用：在GC时一定会被GC回收。

虚引用：由于虚引用只是用来得知对象是否被GC。

## JVM的栈溢出（StackOverflow）

a、线程请求的栈深度大于虚拟机允许的最大深度 StackOverflowError

b、虚拟机在扩展栈深度时，无法申请到足够的内存空间 OutOfMemoryError

## JVM的结构以及分区

**JVM包括了两个子系统和两个组件，分别是Class loader(类装载)、Execution engine(执行引擎)；Runtimely data area（运行时数据区）、Native Interface（本地接口）。**

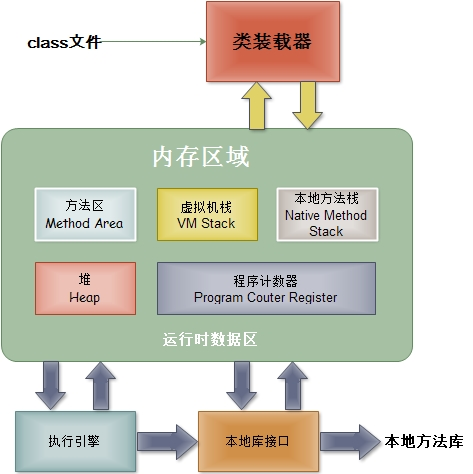
Class loader(类装载)：根据给定的全限定名类名(如：java.lang.Object)来装载class文件到Runtime data area中的method area。

Execution engine（执行引擎）：执行classes中的指令。

Native Interface(本地接口)：与native libraries交互，是其它编程语言交互的接口。

Runtime data area(运行时数据区域)：这就是我们常说的JVM的内存。

**JVM内存区域主要包括：程序计数器(Program Counter)，Java堆(Heap)，Java虚拟机栈(Stack)，本地方法栈(Native Stack)，方法区(Method Area)**



### 程序计数器

程序计数器（Program Counter Register）：当前线程所执行的字节码的行号指示器，字节码解析器的工作是通过改变这个计数器的值，来选取下一条需要执行的字节码指令，分支、循环、跳转、异常处理、线程恢复等基础功能，都需要依赖这个计数器来完成；

### Java 堆

Java 堆（Java Heap）：Java 虚拟机中内存最大的一块，是被所有线程共享的，几乎所有的对象实例都在这里分配内存；

### 方法区

方法区（Methed Area）：用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译后的代码等数据。

### 本地方法栈

本地方法栈（Native Method Stack）：与虚拟机栈的作用是一样的，只不过虚拟机栈是服务 Java 方法的，而本地方法栈是为虚拟机调用 Native 方法服务的；

### JAVA虚拟机栈

虚拟机栈描述Java方法执行的内存模型，每个方法被执行时都会创建一个栈帧(Stack Frame)，栈帧会利用局部变量数组存储局部变量(Local Variables)，操作栈(Operand Stack)，方法出口(Return Value)，动态连接(Current Class Constant Pool Reference)等信息。

对Java虚拟机栈这个区域，Java虚拟机规范规定了两种异常：

**线程请求的栈深度大于虚拟机所允许的深度，抛出StackOverFlow异常。**

**对于支持动态扩展的虚拟机，当扩展无法申请到足够的内存时会抛出OutOfMemory异常。**

## 堆栈的区别？

物理地址

堆的物理地址分配对对象是不连续的。因此性能慢些

栈使用的是数据结构中的栈，先进后出的原则，物理地址分配是连续的。所以性能快。

存放的内容

**堆存放的是对象的实例和数组。因此该区更关注的是数据的存储**

**栈存放：局部变量，操作数栈，返回结果。该区更关注的是程序方法的执行。**

程序的可见度

堆对于整个应用程序都是共享、可见的。

栈只对于线程是可见的。所以也是线程私有。他的生命周期和线程相同。

## JVM的线程

**强引用**：只要有引用指向它，就不会被回收。当内存空间不足时，Java虚拟机宁愿抛出OutOfMemoryError错误；也不会回收它。

如果强引用对象不使用时，需要 strongReference = null;从而使GC能够回收。

**软引用：**内存空间不够时，软引用会被回收。主要用来做缓存

**弱引用：**在垃圾回收器线程扫描它所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。

**虚引用：**虚引用顾名思义，就是形同虚设，使用get也无法获取到虚引用的值。虚引用必须和引用队列(ReferenceQueue)联合使用。

# 多线程

## wait&sleep

sleep来自Thread类，wait来自Object类，他们都会释放CPU

sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可以使用同步控制块或者方法。

## 锁

Java中有两种加锁的方式：一种是用**synchronized**关键字，另一种是用**Lock接口**的实现类。synchronized是一种非公平锁，但是并没有任何办法使其变成公平锁。

公平锁：当锁释放的时候，先申请的线程先得到锁

### 乐观锁：

乐观锁（Optimistic Lock）, 就是很乐观，每次去拿数据的时候都认为别人不会修改。所以不会上锁，**不会上锁**！但是如果想要更新数据，则会在**更新前检查在读取至更新这段时间别人有没有修改过这个数据**。如果修改过，则重新读取，再次尝试更新，循环上述步骤直到更新成功（当然也允许更新失败的线程放弃操作）。

### 悲观锁

悲观锁（Pessimistic Lock）, 就是很悲观，每次去拿数据的时候都认为别人会修改。所以**每次在拿数据的时候都会上锁。**这样别人想拿数据就被挡住，直到悲观锁被释放。

### 自旋锁

**线程获取不到锁就会一直尝试获取，直到成功。都属于悲观锁**

### 互斥锁

**线程获取不到锁就会wait，释放CPU给其他线程，此时发生阻塞 都属于悲观锁**

### 死锁：

死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

## 线程池

参数：

1、最大线程数：maximumPoolSize 线程池最大线程数量，等待队列满的时候，会创建一个新的线程来处理该任务。

2、核心线程数：corePoolSize 线程池核心线程大小

3、keepAliveTime 空闲线程存活时间：一个线程如果处于空闲状态，并且当前的线程数量大于corePoolSize，那么在指定时间后，这个空闲线程会被销毁，这里的指定时间由keepAliveTime来设定

4、unit 空闲线程存活时间单位

5、workQueue：保存任务的队列，当池中线程数大于corePoolSize时，新来的任务保存到该队列。

# tips:

JAVA中所有类的父类是object类，它是java的根类