## 1.Cookie和session

http无状态协议，无法知道是哪个用户访问了网站，cookie,session机制就是为了区分用户

### 区别

Session存储在服务器，cookie存储在客户端

Session放在服务器上会占用服务器资源

Cookie放在本地，传输的是明码信，相对不安全Session传输的只有SessionID是加密的相对安全

Session可以存储任意数据对象，cookie纸存储string类型对象

Cookie可以保留相当长时间，直到记录被删或到失效日期。Session对象在用户结束会话后就自动清除

### 原理

Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一服务器，是在客户端保持状态的方案

用户登录成功后服务器给用户发放一个sessionID,客户端接受后将sessionID存在本地cookie

用户再次访问页面就会携带sessionID,服务器就会更具这个sessionID判断是否有相应的键值

## 2. **HTTP 状态码**



1消息

2成功

3重定向

4请求错误

5服务器错误

301—》请求的资源已被永久的移动到新URL

302—》请求的资源被零食移动到新的URL

400—》bad request请求参数有误

401—》用户没有认证

403—》服务器理解请求但拒绝执行该请求

404—》请求的资源在服务器上未发现

405—》请求的方法不被允许

500—》internet server error 服务器内在错误代码出现错误

503—》服务器过载，无法处理该请求

505—》服务器不支持请求使用的http版本

## 3.char和varChar的区别

 char是固定长度的字符类型，分配多少空间，就占用多长空间。 Varchar是可变长度的字符类型，内容有多大就占用多大的空间，能有效节省空间

## 4. **isset() 和 empty() 区别**

var\_dump(isset($a)); // TRUE var\_dump(isset($a, $b)); // TRUE

Isset判断变量是否存在，可以传入多个变量，若其中一个变量不存在则返回假，empty判断变量是否为空，只可传一个变量，如果为空为假则返回真。

## 5. 匹配URL的正则表达式

Permatch(正则表达式，字符串)字符串返回匹配次数0(不匹配) or 1

‘/^(https?|ftps?):\/\/(www)\.([^\.\/]+)\.(com|cn|org)(\/[\w-\.\/\?\%\&\=]\*)?/i’

## 6. php数组相关的函数

array()----创建数组

array\_combine()----通过合并两个数组来创建一个新数组

range()----创建并返回一个包含指定范围的元素的数组

compact()----建立一个数组

array\_chunk()----将一个数组分割成多个

array\_merge()----把两个或多个数组合并成一个数组

array\_slice()----在数组中根据条件取出一段值

array\_diff()----返回两个数组的差集数组

array\_intersect()----计算数组的交集

array\_search()----在数组中搜索给定的值

array\_splice()----移除数组的一部分且替代它

array\_key\_exists()----判断某个数组中是否存在指定的key

shuffle()----把数组中的元素按随机顺序重新排列

array\_flip()----交换数组中的键和值

array\_reverse()----将原数组中的元素顺序翻转，创建新的数组并返回

array\_unique()----移除数组中重复的值

## 7.缓存

见G:\桌面\学习文件\后端\backEnd\缓存

## 8. 23种设计模式

见G:\桌面\学习文件\后端\backEnd\php后端\PHP基础+面试\设计模式

## 9.mysql

见G:\桌面\学习文件\后端\backEnd\php后端\PHP基础+面试

## 10. javascrip与Jquery相关知识

用javascript取得一个input的值？取得一个input的属性？

|  |  |
| --- | --- |
| 1  2 | document.getElementById(‘name’).value;  document.getElementById(‘name’).type; |

用Jquery取得一个input的值？取得一个input的属性？

|  |  |
| --- | --- |
| 1  2 | $(“input[name='aa']“).val();  $(“input[name='aa']“).attr(‘type’); |

见项目实例

js中网页前进和后退的代码

前进: history.forward();=history.go(1);

后退: history.back();=history.go(-1);

Vue页面跳转： this.$router.push({name:'CachetKind'，cachetid:’1231’})

接收参数： this.cachetid=this.$route.params.cachetid

#### 页面跳转

**window.location.href="login.html";**

## 11.ajax和axios使用

#### Ajax

，是指一种创建交互式、快速动态[网页](https://baike.baidu.com/item/%E7%BD%91%E9%A1%B5/99347)应用的网页开发技术，无需重新加载整个网页的情况下，能够更新部分网页的技术。

 $.ajax({

     type : "POST", *//提交方式*

     url : "${pageContext.request.contextPath}/org/doDelete.action",*//路径*

     data : {

      "org.id" : "${org.id}"

     },*//数据，这里使用的是Json格式进行传输*

     success : function(result) {*//返回数据根据结果进行相应的处理*

      if ( result.success ) {

       $("#tipMsg").text("删除数据成功");

       tree.deleteItem("${org.id}", true);

      } else {

       $("#tipMsg").text("删除数据失败");

      }

     }

#### Axios

this.axios({

method: 'post',

url: '/user/12345',

data: {

firstName: 'Fred',

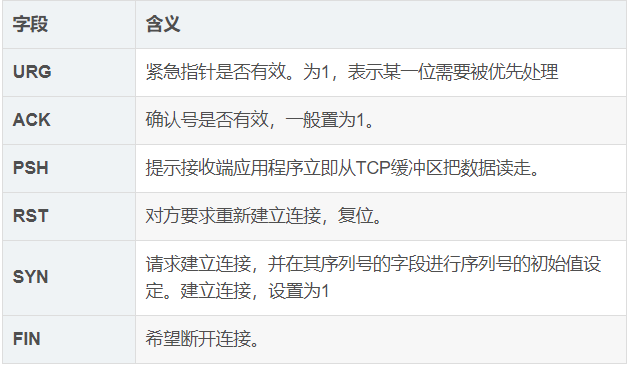
lastName: 'Flintstone'

}

});

## 12.计算机网络网络

### 三次握手



序列号seq：占4个字节，用来标记数据段的顺序，TCP把连接中发送的所有数据字节都编上一个序号，第一个字节的编号由本地随机产生；给字节编上序号后，就给每一个报文段指派一个序号；序列号seq就是这个报文段中的第一个字节的数据编号。

    确认号ack：占4个字节，期待收到对方下一个报文段的第一个数据字节的序号；序列号表示报文段携带数据的第一个字节的编号；而确认号指的是期望接收到下一个字节的编号；因此当前报文段最后一个字节的编号+1即为确认号。

    确认ACK：占1位，仅当ACK=1时，确认号字段才有效。ACK=0时，确认号无效

    同步SYN：连接建立时用于同步序号。当SYN=1，ACK=0时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得SYN=1，ACK=1。因此，SYN=1表示这是一个连接请求，或连接接受报文。SYN这个标志位只有在TCP建产连接时才会被置1，握手完成后SYN标志位被置0。

终止FIN：用来释放一个连接。FIN=1表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接

**Tcp建立的过程**



**第一次握手**: 建立连接。客户端发送连接请求，发送SYN报文，随机生成seq，然后，客户端进入SYN\_SEND状态，等待服务器的确认。

**第二次握手**: 服务器收到客户端的SYN报文段。需要对这个SYN报文段进行确认，发送ACK报文，将ack设置为1（ack值为对方seq+1或者seq+L(数据长度L)）。同时，自己还要发送SYN请求信息，将seq为0。服务器端将上述所有信息一并发送给客户端，此时服务器进入SYN\_RECV状态。

**第三次握手**: 客户端收到服务器的ACK和SYN报文后，进行确认，然后将ack设置为1，seq设置为1，向服务器发送ACK报文段，这个报文段发送完毕以后，客户端和服务器端都进入ESTABLISHED状态，完成TCP三次握手。

#### **为什么需要三次而不是两次**

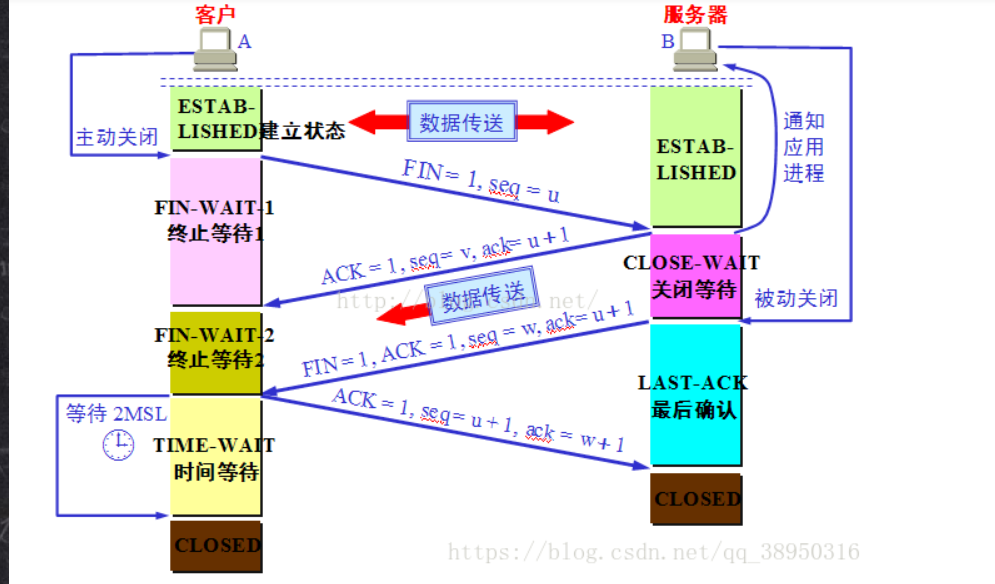
防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。

如果使用的是两次握手建立连接，假设有这样一种场景，客户端发送了第一个请求连接并且没有丢失，只是因为在网络结点中滞留的时间太长了，由于TCP的客户端迟迟没有收到确认报文，以为服务器没有收到，此时重新向服务器发送这条报文，此后客户端和服务器经过两次握手完成连接，传输数据，然后关闭连接。此时此前滞留的那一次请求连接，网络通畅了到达了服务器，这个报文本该是失效的，但是，两次握手的机制将会让客户端和服务器再次建立连接，这将导致不必要的错误和资源的浪费。

果采用两次握手的话，那么服务器认为此时连接已经建立好了。但是当客户端收到这个SYNACK时，如果他并没有发起连接，那么他不会理睬这个SYNACK，就当没事发生过（如果客户端此时正好发起连接，那其实他也不会理睬这个SYNACK，因为确认号不对啊。）。那问题就大了，这时候服务器以为连接好了，向客户端发送数据，而客户端处于CLOSED状态，会丢弃这些包，这样就很浪费了。

### 四次挥手

TCP释放的过程



1）**客户端进程发出连接释放报文，并且停止发送数据**。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，**客户端进入FIN-WAIT-1（终止等待1）状态**。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。

2**）服务器收到连接释放报文，发出确认报文，**ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服**务端就进入了CLOSE-WAIT（关闭等待）状态**。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。

3）**客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态**，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。

4）**服务器将最后的数据发送完毕后，就向客户端发送连接释放报文**，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，**服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。**

5）**客户端收到服务器的连接释放报文后，必须发出确认**，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，**客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2∗∗MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。**

6）**服务器只要收到了客户端发出的确认，立即进入CLOSED状态**。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

#### 为什么客户端最后还要等待2MSL？

第一，保证客户端发送的最后一个ACK报文能够到达服务器，因为这个ACK报文可能丢失，站在服务器的角度看来，我已经发送了FIN+ACK报文请求断开了，客户端还没有给我回应，应该是我发送的请求断开报文它没有收到，于是服务器又会重新发送一次，而客户端就能在这个2MSL时间段内收到这个重传的报文，接着给出回应报文，并且会重启2MSL计时器。

第二，防止类似与“三次握手”中提到了的“已经失效的连接请求报文段”出现在本连接中。客户端发送完最后一个确认报文后，在这个2MSL时间中，就可以使本连接持续的时间内所产生的所有报文段都从网络中消失。这样新的连接中不会出现旧连接的请求报文。

。

#### 为什么建立连接是三次握手，关闭连接确是四次挥手呢？

1. 建立连接的时候， 服务器在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。
2. 而关闭连接时，服务器收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，而自己也未必全部数据都发送给对方了，所以己方可以立即关闭，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送，从而导致多了一次。

## 13.项目所用技术介绍

## 14.php基础

## 15.算法

## 16.数据库优化

优化的查询语句----使用连接（JOIN）来代替子查询(Sub-Queries)

使用索引

表结构优化

分表分库

使用缓存

## 17.安全性攻击

### sql注入

攻击者在HTTP请求中注入恶意的SQL代码，服务器使用参数构建数据库SQL命令时，恶意SQL被一起构造，并在数据库中执行。

用户登录，输入用户名 lianggzone，密码 ‘ or ‘1’=’1 ，如果此时使用参数构造的方式，就会出现

select \* from user where name = 'lianggzone' and password = '' or '1'='1'

不管用户名和密码是什么内容，使查询出来的用户列表不为空。

采用预编译语句集

正则表达式过滤传入参数

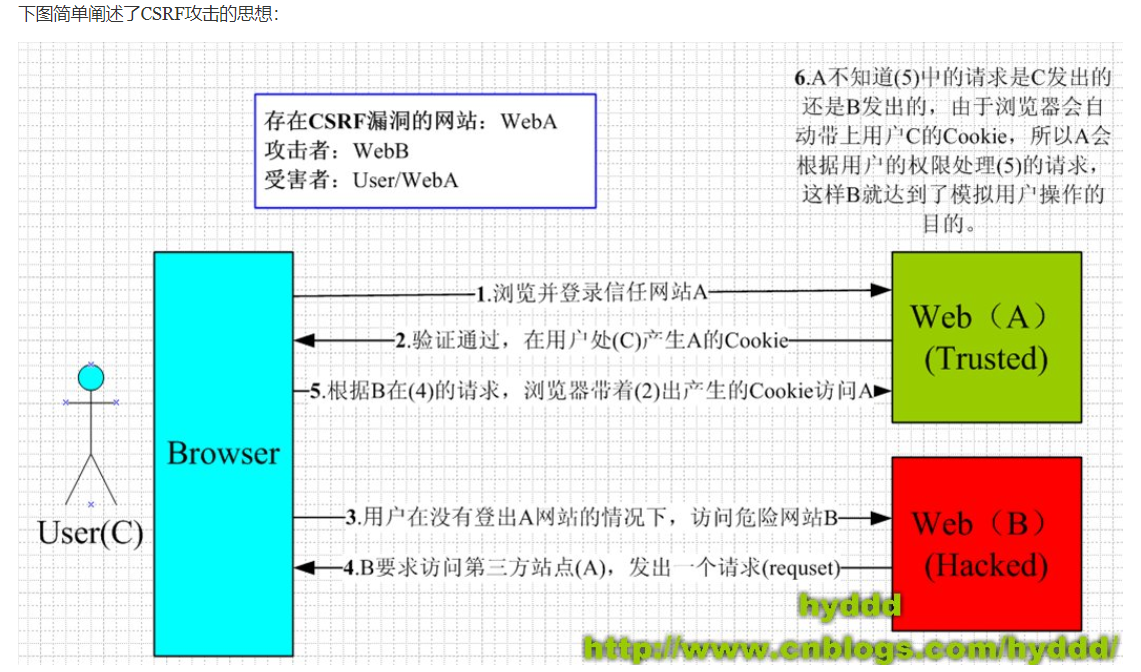
字符串过滤

避免将用户提供的输入直接放入SQL语句中；最好使用准备好的语句和参数化查询，这样更安全。

### CSRF**跨站请求伪造**

参考：<https://www.jianshu.com/p/ffb99fc70646>

攻击者盗用了你的身份，以你的名义发送恶意请求。CSRF能够做的事情包括：以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账......造成的问题包括：个人隐私泄露以及财产安全。



1. 用户打开浏览器，访问受信任银行网站，输入用户名和密码请求登录网站；

2.在用户信息通过验证后，银行网站产生Cookie信息并返回给浏览器，此时用户登录网站成功，可以正常发送请求到网站；

3. 用户未退出银行网站之前，在同一浏览器中，打开一个TAB页访问其他网站B

4. 这时候网站B 已被黑客注入诱导信息，加入是一张图片,图片地址指向

src=”http://bank.example/withdraw?account=bob&amount=1000000&for=黑客

点击之后转账给黑客这个账户

5. 浏览器在接收到这些攻击性代码请求后，根据网站B的请求，在用户不知情的情况下携带Cookie信息，根据用户的Cookie信息以C的权限处理该请求，导致来自黑客请求恶意代码被执行。

#### 防御

**验证 HTTP Referer 字段**

根据 HTTP 协议，在 HTTP 头中有一个字段叫 Referer，它记录了该 HTTP 请求的来源地址。

使用token

**加验证码**

### XSS **跨站脚本攻击**

参考：<https://baijiahao.baidu.com/s?id=1653681087434824406&wfr=spider&for=pc>

XSs击全称跨站脚本攻击，是一种在web应用中的计算机安全漏洞，它允许恶意web用户将代码植入到提供给其它用户使用的页面中。

#### 反射型XSS攻击

 一般是攻击者通过特定手法（如电子邮件），诱使用户去访问一个包含恶意代码的 URL，当受害者点击这些专门设计的链接的时候，恶意代码会直接在受害者主机上的浏览器执行。反射型XSS通常出现在网站的搜索栏、用户登录口等地方，常用来窃取客户端 Cookies 或进行钓鱼欺骗。

#### 存储型XSS攻击

要将XSS代码提交存储在服务器端（数据库，内存，文件系统等），下次请求目标页面时不用再提交XSS代码。当目标用户访问该页面获取数据时，XSS代码会从服务器解析之后加载出来，返回到浏览器做正常的HTML和JS解析执行，XSS攻击就发生了。存储型 XSS 一般出现在网站留言、评论、博客日志等交互处，恶意脚本存储到客户端或者服务端的数据库中。

#### DOM-based 型XSS攻击

DOM 的 XSS 攻击是指通过恶意脚本修改页面的 DOM 结构，是纯粹发生在客户端的攻击。DOM 型 XSS 攻击中，取出和执行恶意代码由浏览器端完成，属于前端 JavaScript 自身的安全漏洞。

#### 预防

**对输入(和URL参数)进行过滤，对输出进行编码**。

1. 对输入内容的特定字符进行编码，例如表示 html标记的 < > 等符号。
2. 对重要的 cookie设置 httpOnly, 防止客户端通过document.cookie读取 cookie，此 HTTP头由服务端设置。
3. 将不可信的值输出 URL参数之前，进行 URLEncode操作，而对于从 URL参数中获取值一定要进行格式检测（比如你需要的时URL，就判读是否满足URL格式）。
4. 不要使用 Eval来解析并运行不确定的数据或代码，对于 JSON解析请使用 JSON.parse() 方法。

 5. 后端接口也应该要做到关键字符过滤的问题。

## 18.高并发解决方案

1、HTML静态化  
效率最高、消耗最小的就是纯静态化的html页面，所以我们尽可能使我们的 网站上的页面采用静态页面来实现，这个最简单的方法其实也是最有效的方法。  
2、图片服务器分离  
把图片单独存储，尽量减少图片等大流量的开销，可以放在一些相关的平台上，如七牛等  
3、数据库集群和库表散列及缓存  
数据库的并发连接为100，一台数据库远远不够，可以从读写分离、主从复制，数据库集群方面来着手。另外尽量减少数据库的访问，可以使用缓存数据库如memcache、redis。  
4、镜像：  
尽量减少下载，可以把不同的请求分发到多个镜像端。  
5、负载均衡：  
Apache的最大并发连接为1500，只能增加服务器，可以从硬件上着手，如F5服务器。当然硬件的成本比较高，我们往往从软件方面着手。

web服务器优化 ：负载均衡   
     流量优化：防盗链处理 将恶意请求屏蔽，  
    前端优化：减少http请求、添加异步请求、启用浏览器缓存和文件压缩、cdn加速、建立独立的图片服务器、  
    服务端优化：  页面静态化、并发处理、队列处理、  
    数据库优化： 数据库缓存、分库分表、分区操作 、读写分离、负载均衡

## 19.http与https的区别

        1、https协议需要到CA申请证书，一般免费证书较少，因而需要一定费用。

        2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl/tls加密传输协议。

        3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

        4、http的连接很简单，是无状态的；HTTPS协议是由SSL/TLS+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

## 20.IP分类

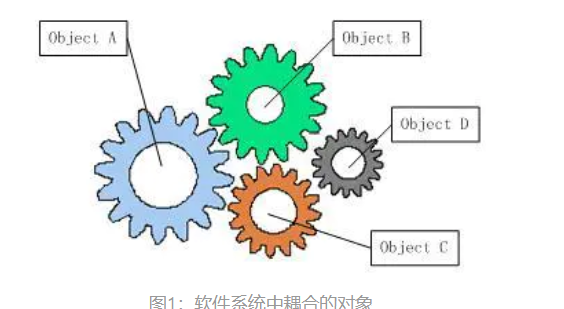
## **21.**面向对象？主要特征是什么？

面向对象是程序的一种设计方式，它利于提高程序的重用性，使程序结构更加清晰。主要特征：封装、继承、多态。

## 22.ioc控制反转

参考：<https://www.jianshu.com/p/07af9dbbbc4b>

在讨论控制反转之前，我们先来看看软件系统中耦合的对象。



从图中可以看到，软件中的对象就像齿轮一样，协同工作，但是互相耦合，一个零件不能正常工作，整个系统就崩溃了。这是一个强耦合的系统。齿轮组中齿轮之间的啮合关系,与软件系统中对象之间的耦合关系非常相似

**控制反转（Inversion of Control）**是一种是面向对象编程中的一种设计原则，用来减低计算机代码之间的耦合度。其基本思想是：借助于“第三方”实现具有依赖关系的对象之间的解耦。

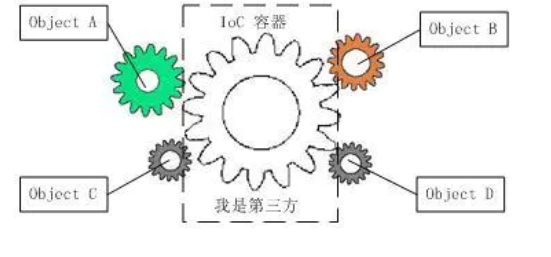


图2：IoC解耦过程

由于引进了中间位置的“第三方”，也就是IOC容器，使得A、B、C、D这4个对象没有了耦合关系，齿轮之间的传动全部依靠“第三方”了，全部对象的控制权全部上缴给“第三方”IOC容器，所以，IOC容器成了整个系统的关键核心，它起到了一种类似“粘合剂”的作用，把系统中的所有对象粘合在一起发挥作用，如果没有这个“粘合剂”，对象与对象之间会彼此失去联系，这就是有人把IOC容器比喻成“粘合剂”的由来。

## 23依赖注入

**依赖注入**就是将实例变量传入到一个对象中去

如果在 Class A 中，有 Class B 的实例，则称 Class A 对 Class B 有一个依赖。例如下面类 Human 中用到一个 Father 对象，我们就说类 Human 对类 Father 有一个依赖。

public class Human {

...

Father father;

...

public Human() {

father = new Father();

}

}

依赖注入主要有两个好处：

1. 解耦，将依赖之间解耦。
2. 因为已经解耦，所以方便做单元测试，尤其是 Mock 测试。

## 24.laravel与tp的比较

ThinkPhp简单、快速，基于MVC和面向对象，易用性较高，是一款简洁实用的轻量级框架。

Lavavel一直秉承着优雅的原则，很好的支持了composer，实现了更丰富的扩展，社区文档活跃，相较于TP，Lavavel更庞大，安全性也更高，更适合开发大中型项目，被称为“巨匠型开发框架”。

## 25.get和post的区别

参考：<https://baijiahao.baidu.com/s?id=1664932291911157034&wfr=spider&for=pc>

**GET产生一个TCP数据包;POST产生两个TCP数据包。**

于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200(返回数据);

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok(返回数据)。

**GET把参数包含在URL中，POST通过request body传递参数**

**GET请求在URL中传送的参数是有长度限制的，而POST没有**

**Post请求相对安全**