JavaWEB面试题

1.说下原生 jdbc 操作数据库流程？（2017-11-25-wzz）

第一步： Class.forName()加载数据库连接驱动；
第二步： DriverManager.getConnection()获取数据连接对象;

第三步：根据 SQL 获取 sql 会话对象，有 2 种方式 Statement、 PreparedStatement ;
第四步：执行 SQL 处理结果集，执行 SQL 前如果有参数值就设置参数值 setXXX();
第五步：关闭结果集、关闭会话、关闭连接

2.什么要使用 PreparedStatement？

1、 PreparedStatement 接口继承 Statement， PreparedStatement 实例包含已编译的 SQL 语句，所以其执行
速度要快于 Statement 对象。

2、作为 Statement 的子类，PreparedStatement 继承了 Statement 的所有功能。三种方法 execute、 executeQuery 和 executeUpdate 已被更改以使之不再需要参数

3、在 JDBC 应用中,在任何时候都不要使用 Statement，原因如下：

一、代码的可读性和可维护性.Statement 需要不断地拼接，而 PreparedStatement 不会。
二、 PreparedStatement 尽最大可能提高性能.DB 有缓存机制，相同的预编译语句再次被调用不会再次需要
编译。
三、最重要的一点是极大地提高了安全性.Statement 容易被 SQL 注入，而 PreparedStatementc 传入的内容不会和 sql 语句发生任何匹配关系。

3.关系数据库中连接池的机制是什么？

前提：为数据库连接建立一个缓冲池。
1：从连接池获取或创建可用连接
2：使用完毕之后，把连接返回给连接池
3：在系统关闭前，断开所有连接并释放连接占用的系统资源

4：能够处理无效连接，限制连接池中的连接总数不低于或者不超过某个限定值。

其中有几个概念需要大家理解：
最小连接数是连接池一直保持的数据连接。如果应用程序对数据库连接的使用量不大，将会有大量的数据库连接
资源被浪费掉。

最大连接数是连接池能申请的最大连接数。如果数据连接请求超过此数，后面的数据连接请求将被加入到等待队
列中，这会影响之后的数据库操作。

如果最小连接数与最大连接数相差太大，那么，最先的连接请求将会获利，之后超过最小连接数量的连接请求等
价于建立一个新的数据库连接。不过，这些大于最小连接数的数据库连接在使用完不会马上被释放，它将被放到连接池中等待重复使用或是空闲超时后被释放。

上面的解释，可以这样理解：数据库池连接数量一直保持一个不少于最小连接数的数量，当数量不够时，数据库会
创建一些连接，直到一个最大连接数，之后连接数据库就会等待。

4.http 的长连接和短连接

HTTP 协议有 HTTP/1.0 版本和 HTTP/1.1 版本。 HTTP1.1 默认保持长连接（HTTP persistent connection，也
翻译为持久连接），数据传输完成了保持 TCP 连接不断开（不发 RST 包、不四次握手），等待在同域名下继续用这个通道传输数据；相反的就是短连接。

在 HTTP/1.0 中，默认使用的是短连接。也就是说，浏览器和服务器每进行一次 HTTP 操作，就建立一次连接，任
务结束就中断连接。 从 HTTP/1.1 起，默认使用的是长连接， 用以保持连接特性。

5.HTTP/1.1 与 HTTP/1.0 的区别

1 可扩展性
a) HTTP/1.1 在消息中增加版本号，用于兼容性判断。
b) HTTP/1.1 增加了 OPTIONS 方法，它允许客户端获取一个服务器支持的方法列表。
c) 为了与未来的协议规范兼容， HTTP/1.1 在请求消息中包含了 Upgrade 头域，通过该头域，客户端可以让服务器知道它能够支持的其它备用通信协议，服务器可以据此进行协议切换，使用备用协议与客户端进行通信。

2 缓存
在 HTTP/1.0 中，使用 Expire 头域来判断资源的 fresh 或 stale，并使用条件请求（conditional request）来判断资源是否仍有效。 HTTP/1.1 在 1.0 的基础上加入了一些 cache 的新特性，当缓存对象的 Age 超过 Expire 时变stale 对象， cache 不需要直接抛弃 stale 对象，而是与源服务器进行重新激活（revalidation）。

3 带宽优化
HTTP/1.0 中，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来
了。例如，客户端只需要显示一个文档的部分内容，又比如下载大文件时需要支持断点续传功能，而不是在发生断连后不得不重新下载完整的包。

HTTP/1.1 中在请求消息中引入了 range 头域，它允许只请求资源的某个部分。在响应消息中 Content-Range 头
域声明了返回的这部分对象的偏移值和长度。如果服务器相应地返回了对象所请求范围的内容，则响应码为 206

（Partial Content），它可以防止 Cache 将响应误以为是完整的一个对象。

另外一种情况是请求消息中如果包含比较大的实体内容，但不确定服务器是否能够接收该请求（如是否有权限），
此时若贸然发出带实体的请求，如果被拒绝也会浪费带宽。

HTTP/1.1 加入了一个新的状态码 100（Continue）。客户端事先发送一个只带头域的请求，如果服务器因为权
限拒绝了请求，就回送响应码 401（Unauthorized）；如果服务器接收此请求就回送响应码 100，客户端就可以继续

发送带实体的完整请求了。注意， HTTP/1.0 的客户端不支持 100 响应码。但可以让客户端在请求消息中加入 Expect头域，并将它的值设置为 100-continue。

节省带宽资源的一个非常有效的做法就是压缩要传送的数据。 Content-Encoding 是对消息进行端到端（end-toend）的编码，它可能是资源在服务器上保存的固有格式（如 jpeg 图片格式）；在请求消息中加入 Accept-Encoding
头域，它可以告诉服务器客户端能够解码的编码方式

4 长连接
HTTP/1.0 规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个 TCP 连接，服务
器完成请求处理后立即断开 TCP 连接，服务器不跟踪每个客户也不记录过去的请求。此外，由于大多数网页的流量都比较小，一次 TCP 连接很少能通过 slow-start 区，不利于提高带宽利用率。

HTTP 1.1 支持长连接（PersistentConnection）和请求的流水线（Pipelining）处理，在一个 TCP 连接上可以传
送多个 HTTP 请求和响应，减少了建立和关闭连接的消耗和延迟。例如：一个包含有许多图像的网页文件的多个请求和应答可以在一个连接中传输，但每个单独的网页文件的请求和应答仍然需要使用各自的连接。

HTTP 1.1 还允许客户端不用等待上一次请求结果返回，就可以发出下一次请求，但服务器端必须按照接收到客户
端请求的先后顺序依次回送响应结果，以保证客户端能够区分出每次请求的响应内容，这样也显著地减少了整个下载过程所需要的时间

5 消息传递
HTTP 消息中可以包含任意长度的实体，通常它们使用 Content-Length 来给出消息结束标志。但是，对于很多动
态产生的响应，只能通过缓冲完整的消息来判断消息的大小，但这样做会加大延迟。如果不使用长连接，还可以通过连接关闭的信号来判定一个消息的结束。

HTTP/1.1 中引入了 Chunkedtransfer-coding 来解决上面这个问题，发送方将消息分割成若干个任意大小的数据块，每个数据块在发送时都会附上块的长度，最后用一个零长度的块作为消息结束的标志。这种方法允许发送方只
缓冲消息的一个片段，避免缓冲整个消息带来的过载

在 HTTP/1.0 中，有一个 Content-MD5 的头域，要计算这个头域需要发送方缓冲完整个消息后才能进行。而
HTTP/1.1 中，采用 chunked 分块传递的消息在最后一个块（零长度）结束之后会再传递一个拖尾（trailer），它包含一个或多个头域，这些头域是发送方在传递完所有块之后再计算出值的。发送方会在消息中包含一个 Trailer 头域告诉接收方这个拖尾的存在。

6 Host 头域
在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。

但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个 IP 地址。

HTTP1.1 的请求消息和响应消息都应支持 Host 头域，且请求消息中如果没有 Host 头域会报告一个错误（400
Bad Request）。此外，服务器应该接受以绝对路径标记的资源请求。

7 错误提示
HTTP/1.0 中只定义了 16 个状态响应码，对错误或警告的提示不够具体。 HTTP/1.1 引入了一个 Warning 头域，
增加对错误或警告信息的描述。

此外，在 HTTP/1.1 中新增了 24 个状态响应码，如 409（Conflict）表示请求的资源与资源的当前状态发生冲突；
410（Gone）表示服务器上的某个资源被永久性的删除

6.http 常见的状态码有哪些？

200 OK //客户端请求成功
301 Moved Permanently（永久移除)，请求的 URL 已移走。 Response 中应该包含一个 Location URL, 说明资
源现在所处的位置
302 found 重定向
400 Bad Request //客户端请求有语法错误， 不能被服务器所理解
401 Unauthorized //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务
404 Not Found //请求资源不存在， eg：输入了错误的 URL

500 Internal Server Error //服务器发生不可预期的错误
503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

7.GET 和 POST 的区别？

从表面现像上面看 GET 和 POST 的区别：

1. GET 请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以?分割 URL 和传输数据，参数之间以&相连，如： login.action?name=zhagnsan&password=123456。 POST 把提交的数据则放置在是 HTTP 包的包体中。
2. GET 方式提交的数据最多只能是 1024 字节，理论上 POST 没有限制，可传较大量的数据。其实这样说是错误
   的，不准确的： GET 方式提交的数据最多只能是 1024 字节"，因为 GET 是通过 URL 提交数据，那么 GET 可提交的数据量就跟URL 的长度有直接关系了。而实际上， URL 不存在参数上限的问题， HTTP 协议规范没有对 URL 长度进行限制。这个限制是特定的浏览器及服务器对它的限制。IE对URL长度的限制是2083 字节(2K+35)。对于其他浏览器，如 Netscape、FireFox 等，理论上没有长度限制，其限制取决于操作系统的支持。
3. POST 的安全性要比 GET 的安全性高。注意：这里所说的安全性和上面 GET 提到的“安全”不是同个概念。上
   面“安全”的含义仅仅是不作数据修改，而这里安全的含义是真正的 Security 的含义，比如：通过 GET 提交数据，用户名和密码将明文出现在 URL 上，因为(1)登录页面有可能被浏览器缓存， (2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用 GET 提交数据还可能会造成 Cross-site request forgery 攻击。Get 是向服务器发索取数据的一种请求，

而 Post 是向服务器提交数据的一种请求，在 FORM（表单）中， Method默认为"GET"，实质上， GET 和 POST 只是发送机制不同，并不是一个取一个发！

8.http 中重定向和请求转发的区别？

本质区别： 转发是服务器行为，重定向是客户端行为。
重定向特点：两次请求，浏览器地址发生变化，可以访问自己 web 之外的资源，传输的数据会丢失。
请求转发特点：一次强求，浏览器地址不变，访问的是自己本身的 web 资源，传输的数据不会丢失

9.Cookie 和 Session 的区别

Cookie 是 web 服务器发送给浏览器的一块信息，浏览器会在本地一个文件中给每个 web 服务器存储 cookie。
以后浏览器再给特定的 web 服务器发送请求时，同时会发送所有为该服务器存储的 cookie。

Session 是存储在 web 服务器端的一块信息。 session 对象存储特定用户会话所需的属性及配置信息。当用户在
应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下。

Cookie 和 session 的不同点：

1、无论客户端做怎样的设置， session 都能够正常工作。当客户端禁用 cookie 时将无法使用 cookie。

2、在存储的数据量方面： session 能够存储任意的 java 对象， cookie 只能存储 String 类型的对象

10.session 共享怎么做的（分布式如何实现 session 共享）

问题描述：一个用户在登录成功以后会把用户信息存储在 session 当中，这时 session 所在服务器为 server1，那
么用户在 session 失效之前如果再次使用 app，那么可能会被路由到 server2，这时问题来了， server 没有该用户的session，所以需要用户重新登录，这时的用户体验会非常不好，所以我们想如何实现多台 server 之间共享 session，让用户状态得以保存。

1、服务器实现的 session 复制或 session 共享，这类型的共享 session 是和服务器紧密相关的，比如 webSphere或 JBOSS 在搭建集群时候可以配置实现 session 复制或 session 共享，但是这种方式有一个致命的缺点，就是不好扩展和移植，比如我们更换服务器，那么就要修改服务器配置。

2、利用成熟的技术做session复制，比如12306使用的gemfire，比如常见的内存数据库如redis或 memorycache，这类方案虽然比较普适，但是严重依赖于第三方，这样当第三方服务器出现问题的时候，那么将是应用的灾难。

3、将 session 维护在客户端，很容易想到就是利用 cookie，但是客户端存在风险，数据不安全，而且可以存放的 数据量比较小，所以将 session 维护在客户端还要对 session 中的信息加密。

我们实现的方案可以说是第二种方案和第三种方案的合体，可以利用 gemfire 实现 session 复制共享，还可以将
session 维护在 redis 中实现 session 共享，同时可以将 session 维护在客户端的 cookie 中，但是前提是数据要加密。

这三种方式可以迅速切换，而不影响应用正常执行。我们在实践中，首选 gemfire 或者 redis 作为 session 共享的载体，一旦 session 不稳定出现问题的时候，可以紧急切换 cookie 维护 session 作为备用，不影响应用提供服务。

这里主要讲解 redis 和 cookie 方案， gemfire 比较复杂大家可以自行查看 gemfire 工作原理。利用 redis 做
session 共享，首先需要与业务逻辑代码解耦，不然 session 共享将没有意义，其次支持动态切换到客户端 cookie 模式。 redis 的方案是，重写服务器中的 HttpSession 和 HttpServletRequest，首先实现 HttpSession 接口，重写 session的所有方法，将 session 以 hash 值的方式存在 redis 中，一个 session 的 key 就是 sessionID， setAtrribute 重写之后就是更新 redis 中的数据， getAttribute 重写之后就是获取 redis 中的数据，等等需要将 HttpSession 的接口一一实现

实现了 HttpSesson，那么我们先将该 session 类叫做 MySession（当然实践中不是这么命名的），当 MySession
出现之后问题才开始，怎么能在不影响业务逻辑代码的情况下，还能让原本的 request.getSession（）获取到的是MySession，而不是服务器原生的 session。这里，我决定重写服务器的 HttpServletRequet，这里先称为 MyRequest，但是这可不是单纯的重写，我需要在原生的 request 基础上重写，于是我决定在 filter 中，实现 request 的偷梁换柱，我的思路是这样的， MyRequest 的构建器，必须以 request 作为参数，于是我在 filter 中将服务器原生的 request（也有 可能 是框 架封装过的 request ），当 做参数 new 出 来一 个 MyRequest ，并 且 MyRequest 也 实现 了 HttpServletRequest 接口，其实就是对原生 request 的一个增强，这里主要重写了几个 request 的方法，但是最重要的是重写了 request.getSession()，写到这里大家应该都明白为什么重写这个方法了吧，当然是为了获取 MySession，于是这样就在filter中，偷偷的将原生的request换成MyRequest了，然后再将替换过的request传入chan.doFilter()，这样 filter 时候的代码都使用的是 MyRequest 了，同时对业务代码是透明的，业务代码获取 session 的方法仍然是request.getSession（），但其实获取到的已经是 MySession 了，这样对 session 的操作已经变成了对 redis 的操作。

这样实现的好处有两个，第一开发人员不需要对 session 共享做任何关注， session 共享对用户是透明的；第二， filter是可配置的，通过 filter 的方式可以将 session 共享做成一项可插拔的功能，没有任何侵入性。

这个时候已经实现了一套可插拔的 session 共享的框架了，但是我们想到如果 redis 服务出了问题，这时我们该怎
么办呢，于是我们延续 redis 的想法，想到可以将 session 维护在客户端内（加密的 cookie），当然实现方法还是一样的，我们重写 HttpSession 接口，实现其所有方法，比如 setAttribute 就是写入 cookie， getAttribute 就是读取cookie，我们可以将重写的 session 称作 MySession2，这时怎么让开发人员透明的获取到 MySession2 呢，实现方法还是在 filter 内偷梁换柱，在 MyRequest 加一个判断，读取 sessionType 配置，如果 sessionType 是 redis 的，那么 getSession 的时候获取到的是 MySession，如果 sessionType 是 coolie 的，那么 getSession 的时候获取到的是MySession2，以此类推，用同样的方法就可以获取到 MySession 3,4,5,6 等等。

这样两种方式都有了，那么我们怎实现两种 session 共享方式的快速切换呢，刚刚我提到一个 sessionType，这
是用来决定获取到session的类型的，只要变换sessionType就能实现两种session共享方式的切换，但是sessionType必须对所有的服务器都是一致的，如果不一致那将会出现比较严重的问题，我们目前是将 sessionType 维护在环境变量里，如果要切换 sessionType 就要重启每一台服务器，完成 session 共享的转换，但是当服务器太多的时候将是一种灾难。而且重启服务意味着服务的中断，所以这样的方式只适合服务器规模比较小，而且用户量比较少的情况，当服务器太多的时候，务必需要一种协调技术，能够让服务器能够及时获取切换的通知。基于这样的原因，我们选用zookeeper 作为配置平台，每一台服务器都会订阅 zookeeper 上的配置，当我们切换 sessionType 之后，所有服务器都会订阅到修改之后的配置，那么切换就会立即生效，当然可能会有短暂的时间延迟，但这是可以接受的。

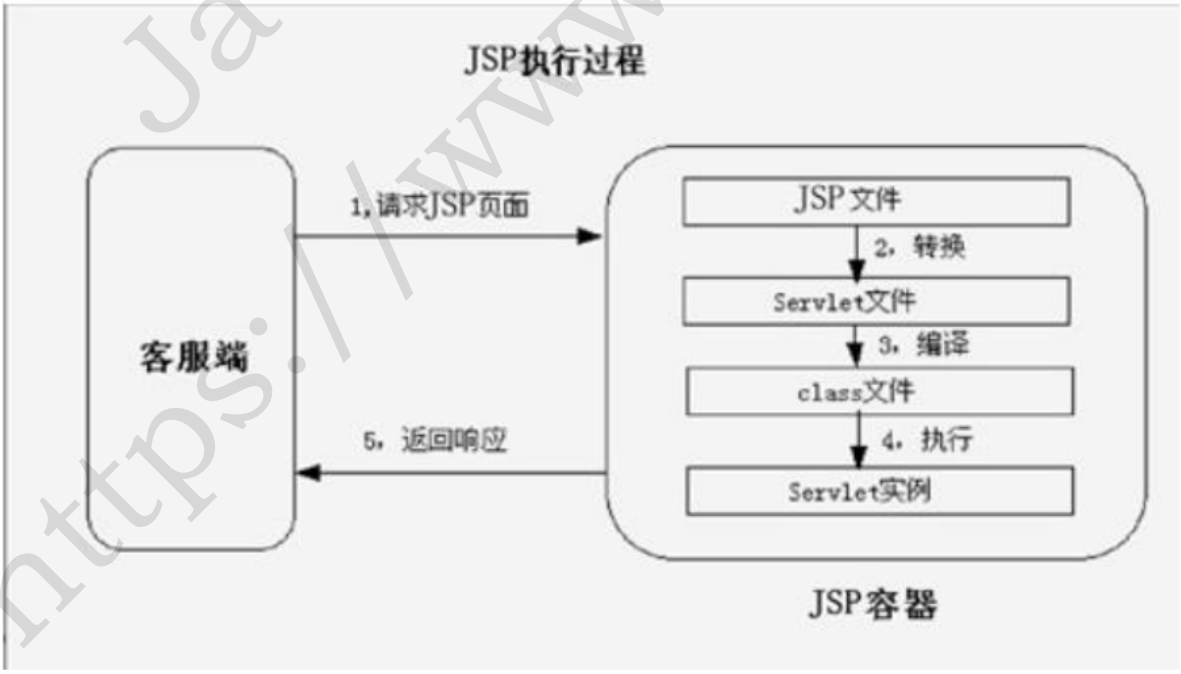
11.在单点登录中，如果 cookie 被禁用了怎么办？

单点登录的原理是后端生成一个 session ID，然后设置到 cookie，后面的所有请求浏览器都会带上 cookie，
然后服务端从 cookie 里获取 session ID，再查询到用户信息。所以，保持登录的关键不是 cookie，而是通过
cookie 保存和传输的 session ID，其本质是能获取用户信息的数据。除了 cookie，还通常使用 HTTP 请求头来传
输。但是这个请求头浏览器不会像 cookie 一样自动携带，需要手工处理。

12.什么是 jsp，什么是 Servlet？ jsp 和 Servlet 有什么区别？

jsp 本质上就是一个 Servlet，它是 Servlet 的一种特殊形式（由 SUN 公司推出），每个 jsp 页面都是一个 servlet
实例。
Servlet 是由 Java 提供用于开发 web 服务器应用程序的一个组件，运行在服务端，由 servlet 容器管理，用来生
成动态内容。一个 servlet 实例是实现了特殊接口 Servlet 的 Java 类，所有自定义的 servlet 均必须实现 Servlet 接口。

区别：
jsp 是 html 页面中内嵌的 Java 代码，侧重页面显示；
Servlet 是 html 代码和 Java 代码分离，侧重逻辑控制， mvc 设计思想中 jsp 位于视图层， servlet 位于控制层
Jsp 运行机制：如下图



JVM 只能识别 Java 类，并不能识别 jsp 代码！ web 容器收到以.jsp 为扩展名的 url 请求时，会将访问请求交给
tomcat 中 jsp 引擎处理，每个 jsp 页面第一次被访问时， jsp 引擎将 jsp 代码解释为一个 servlet 源程序，接着编译servlet 源程序生成.class 文件，再有 web 容器 servlet 引擎去装载执行 servlet 程序，实现页面交互。

13.jsp 有哪些域对象和内置对象及他们的作用？

四大域对象：
（1）pageContext page 域-指当前页面，在当前 jsp 页面有效，跳到其它页面失效
（2）request request 域-指一次请求范围内有效，从 http 请求到服务器处理结束，返回响应的整个过程。
在这个过程中使用 forward（请求转发）方式跳转多个 jsp，在这些页面里你都可以使用这个变量
（3）session session 域-指当前会话有效范围，浏览器从打开到关闭过程中，转发、重定向均可以使用
（4）application context 域-指只能在同一个 web 中使用，服务器未关闭或者重启，数据就有效
九大内置对象：

	生命周期	作用域	使用情况
Request	一次请求	只在 Jsp 页面内有效	用于接受通过 HTTP 协议传送到服务器 的数据（包括头信息、系统信息、请 求方式以及请求参数等）。

	生命周期	作用域	使用情况
Reponse	一次响应	只在 Jsp 页面内有效	表示服务器端对客户端的回应。主要 用于设置头信息、跳转、 Cookie
Session	从存入数据开始，默认闲置 30 分钟后失 效	会话内有效	用于存储特定的用户会话所需的信息
Out	http://www.cnblogs.com/leirenyuan/p/601 6063.html		用于在 Web 浏览器内输出信息，并且 管理应用服务器上的输出缓冲区
PageContext	详细了解： http://www.cnblogs.com/leirenyuan/p/601 6063.html		用于存取其他隐含对象，如 request、reponse、 session、 application 等对 象。（实际上， pageContext 对象提供 了对 JSP 页面所有的对象及命名空间的 访问
Page	http://www.cnblogs.com/leirenyuan/p/601 6063.html		page 对象代表 JSP 本身（对应 this ），只有在 JSP 页面内才是合法的
Exception	http://www.cnblogs.com/leirenyuan/p/601 6063.html		显示异常信息， 必须在 page 指令中设 定<%@ page isErrorPage="true" %>才能 使用，在一般的 JSP 页面中使用该对象 将无法编译 JSP 文件
Application	服务器启动发送第一个请求时就产生了 Application 对象，直到服务器关闭。		用于存储和访问来自任何页面的变量 所有的用户分享一个 Application 对象
Config	http://www.cnblogs.com/leirenyuan/p/601 6063.html		取得服务器的配置信息

14.什么是 xml，使用 xml 的优缺点， xml 的解析器有哪几种，分别有什么区别？

xml 是一种可扩展性标记语言，支持自定义标签（使用前必须预定义）使用 DTD 和 XML Schema 标准化 XML 结
构

优点：用于配置文件，格式统一，符合标准；用于在互不兼容的系统间交互数据，共享数据方便；

缺点： xml 文件格式复杂，数据传输占流量，服务端和客户端解析 xml 文件占用大量资源且不易维护
Xml 常用解析器有 2 种，分别是： DOM 和 SAX;

主要区别在于它们解析 xml 文档的方式不同。使用 DOM 解析， xml 文档以 DOM树形结构加载入内存，而 SAX 采用的是事件模型，

15.谈谈你对 ajax 的认识？

Ajax 是一种创建交互式网页应用的的网页开发技术； Asynchronous JavaScript and XML"的缩写。

Ajax 的优势：
通过异步模式，提升了用户体验。
优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用。
Ajax 引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。

Ajax 的最大特点：
可以实现局部刷新，在不更新整个页面的前提下维护数据，提升用户体验度。

> 注意：ajax 在实际项目开发中使用率非常高（牢固掌握），针对 ajax 的详细描述：

16.jsonp 原理

JavaScript 是一种在 Web 开发中经常使用的前端动态脚本技术。在 JavaScript 中，有一个很重要的安全性限制，被称为"Same-Origin Policy"（同源策略）。这一策略对于 JavaScript 代码能够访问的页面内容做了很重要的限制，即 JavaScript 只能访问与包含它的文档在同一域下的内容。

JavaScript 这个安全策略在进行多 iframe 或多窗口编程、以及 Ajax 编程时显得尤为重要。根据这个策略，在 baidu.com 下的页面中包含的 JavaScript 代码，不能访问在 google.com 域名下的页面内容；甚至不同的子域名之间的页面也不能通过 JavaScript 代码互相访问。对于 Ajax 的影响在于，通过 XMLHttpRequest 实现的Ajax 请求，不能向不同的域提交请求，例如，在 abc.example.com 下的页面，不能向 def.example.com 提交Ajax 请求，等等。

然而，当进行一些比较深入的前端编程的时候，不可避免地需要进行跨域操作，这时候"同源策略"就显得过于苛刻。 JSONP 跨域 GET 请求是一个常用的解决方案，下面我们来看一下 JSONP 跨域是如何实现的，并且探讨下 JSONP 跨域的原理。

jsonp 的最基本的原理是：动态添加一个标签， 使用 script 标签的 src 属性没有跨域的限制的特点实现跨域。首先在客户端注册一个 callback, 然后把 callback 的名字传给服务器。此时，服务器先生成 json 数据。 然后以 javascript 语法的方式，生成一个 function , function 名字就是传递上来的参数 jsonp。最后将json 数据直接以入参的方式，放置到 function 中，这样就生成了一段 js 语法的文档，返回给客户端。

客户端浏览器，解析 script 标签，并执行返回的 javascript 文档，此时数据作为参数，传入到了客户端预先定义
好的 callback 函数里。

17.谈谈你对 restful 的理解以及在项目中的使用？

一种软件架构风格、设计风格，而不是标准，只是提供了一组设计原则和约束条件。它主要用于客户端和服务器交
互类的软件。 REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

它结构清晰、符合标准、易于理解、扩展方便，所以正得到越来越多网站的采用。

给大家推荐如下一篇博客，该博客从多个维度讲解了什么是 Restful 并且给了 Restful 风格样式的 API 接口。

18.什么是 webService？

WebService 是一种跨编程语言和跨操作系统平台的远程调用技术。所谓跨编程语言和跨操作平台，就是说服务端程序采用 java 编写，客户端程序则可以采用其他编程语言编写，反之亦然！跨操作系统平台则是指服务端程序和
户端程序可以在不同的操作系统上。

19.常见的远程调用技术

RMI 是 java 语言本身提供的远程通讯协议，稳定高效，是 EJB 的基础。但它只能用于 JAVA 程序之间的通讯。
Hessian 和 Burlap 是 caucho 公司提供的开源协议，基于 HTTP 传输，服务端不用开防火墙端口。协议的规范公
开，可以用于任意语言。跨平台有点小问题。

Httpinvoker 是 SpringFramework 提供的远程通讯协议，只能用于 JAVA 程序间的通讯，且服务端和客户端必须
使用 SpringFramework。

Web service 是连接异构系统或异构语言的首选协议，它使用 SOAP 形式通讯，可以用于任何语言，目前的许多
开发工具对其的支持也很好。

效率相比： RMI > Httpinvoker >= Hessian >> Burlap >> web service。