# 1.第二阶段面试题

## 描述Servlet调用过程？

**答案：**

（1）在浏览器输入地址，浏览器先去查找hosts文件，将主机名翻译为ip地址，如果找不到就再去查询dns服务器将主机名翻译成ip地址。

（2）浏览器根据ip地址和端口号访问服务器，组织http请求信息发送给服务器。

（3）服务器收到请求后首先根据Host请求头判断当前访问的是哪台虚拟主机。

（4）服务器根据http请求头中的请求URI判断当前访问的是哪个web应用。

（5）服务器根据http请求头中的请求URI判断当前访问的是web应用中的哪个web资源。

（6）检查web应用的web.xml文件，如果根据路径找到具体的servlet处理类的全路径名交给该servlet处理,如果找不到就交给缺省servlet处理。

（7）这个过程中浏览器只知道自己发出来http请求，不久就收到了http响应，浏览器不知道也不关心服务器内部是如何处理的。浏览器和服务器之间的关系是非常单纯的，只有HTTP协议。

（8）解析请求、封装RequestResponse对象、创建Servlet、调用Service方法都是服务器自动进行的，开发人员只需要写好Servlet配置进容器中即可，无需操心具体的底层实现。

## 简述Servlet生命周期？

**答案：**

（1）Servlet第一次被访问到时创建对象，创建出来后立即执行init方法执行初始化的操作。

（2）从此以后该对象一直驻留在内存中为后续的对这个Servlet的请求进行服务。

（3）直到服务器关闭或web应用移除出容器时，随着web应用的销毁Servlet对象销毁掉，在销毁之前调用destory方法执行善后工作。

（4）在存活期间，每次对Servlet 的调用都会导致Service方法的执行。

## 什么是http协议？

**答案：**

HTTP协议就是一套基于tcp/ip协议的应用层协议 。简单来说，就是一个基于应用层的通信规范，双方要进行通信，大家都要遵守一个规范，这个规范就是HTTP协议。它规定了客户端（通常是浏览器）和服务器之间的通信方式。

## HTTP协议工作原理？

**答案：**

HTTP协议基于请求响应模型。

一次请求对应一次响应。

首先客户端发送一个请求(request)给服务器，服务器在接收到这个请求后将生成一个响应(response)返回给客户端。

## HTTP协议的特点是什么 ?

**答案：**

1. 它是一个无状态的协议，服务器端在处理相应请求后不会保留任何客户端的信息，每次请求都是独立的
2. 客户端与服务器端的每一次数据交互，都要经过一次请求/响应的过程。
3. 服务器端无法识别能够出发客户端请求的方法。
4. 一个典型的HTTP请求分为 一个请求行 若干请求头 一个空行 实体内容。

## get和post请求的区别？

**答案：**

1. get请求用来从服务器上获得资源，而post是用来向服务器提交数据；
2. get将表单中数据按照name=value的形式，添加到action 所指向的URL 后面，并且两者使用"?"连接，而各个变量之间使用"&"连接；post是将表单中的数据放在HTTP协议的请求头或消息体中，传递到action所指向URL；
3. get传输的数据要受到URL长度限制（1024字节）；而post可以传输大量的数据， POST数据是没有限制的，上传文件通常要使用post方式；
4. 使用get时参数会显示在地址栏上，如果这些数据不是敏感数据，那么可以使用get；对于敏感数据还是应用使用post；
5. get使用MIME类型application/x-www-form-urlencoded的URL编码（也叫百分号编码）文本的格式传递参数，保证被传送的参数由遵循规范的文本组成，例如一个空格的编码是"%20"。
6. Jsp页面中的FORM标签里的method属性为get时调用doGet()，为post时调用doPost()。

## 请求乱码产生的原因？

**答案：**

浏览器用什么码表来打开表单页面就用什么编码来发送数据。当前我们的注册页面指定了用utf-8来打开。

这就决定了浏览器是用utf-8打开的页面，浏览器在提交表单时是用utf-8编码的。

而tomcat默认情况下会使用iso8859-1来进行解码。

我们知道全世界的码表都兼容iso8859-1，所以英文处理是没有问题的。

但是iso8859-1中并没有中文，iso8859-1对于无法处理的字节都使用?替代，所以我们看到的都是？。

## 如何来处理get请求产生的乱码?

**答案：**

由于客户端发送时使用的是utf-8编码而服务器用iso8859-1解码造成了乱码，虽然字符已经乱掉了，但底层的字节仍然是正确的，我们只要将乱码字符getBytes(“iso8859-1”)转换为字节，就是正确的字节，再将这些字节new String(bytes，“utf-8”)按照正确的码表编码，就可以转换回正确的字符了。从而解决了乱码。

## Request生命周期

**答案：**

request对象的生命周期是针对一个客户端(一个浏览器应用程序)的一次请求，当请求完毕之后，request里边的内容也将被释放，一个请求开始时创建，请求结束后销毁。

## 如何处理响应乱码？

**答案：**

通过response.setHeader("Content-Type", "text/html;charset=utf-8")方法，通知服务器发送数据时的码表；通过response.setCharacterEncoding("utf-8")方法，通知浏览器解析时使用的码表。两码相同就不会有乱码了。

response提供了setContentType("text/html;charset=UTF-8")快捷方法，在它的底层，会同时做上面两件事，所以可以一行代码解决response产生的乱码问题。

## 简述ServletContext生命周期？

**答案：**

ServletContext对象代表当前web应用。当服务器启动时，服务器在启动时会依次加载web应用，每一个web应用加载完成后都会创建一个ServletContext对象唯一代表该web应用，这个对象一直存活，直到web应用移除出容器或服务器关闭时，随着应用销毁，ServletContext对象跟着销毁。

## 转发与重定向的比较？

**答案：**

转发是服务器内部资源跳转,重定向是通过302+Location实现浏览器跳转访问。

转发一次请求一次响应,重定向两次请求两次响应。

转发地址栏不发生变化,重定向地址栏会发生变化。

转发之前和转发之后request是一个,重定向之前和之后不是一个request。

## Session生命周期？

**答案：**

当程序第一次调用到request.getSession()代码时,服务器明确的知道了需要用到session了,此时创建session。

如果session超过30分钟(可以在web.xml中配置的)没人使用,服务器认为这个session超时了,销毁session。

明确的调用session.invalidate(),session立即销毁。

服务器被非正常关闭或web应用被移除出容器,此时随着web应用的销毁session销毁.如果是正常关闭,session会被钝化.当下次服务器正常启动时,没有超时的session还会被活化回来。

## session的原理？

**答案：**

session的原理：在服务器第一次调用request.getSession()方法的时候，会在内存中创建一个session对象，此对象具有一个独一无二的id值，此id值将会以cookie（JSESSIONID）的形式发送给浏览器，浏览器以后每次访问都会带着此cookie，服务器就利用此cookie区分浏览器找到对应的session空间。

## cookie与session的区别

**答案：**

cookie数据存放在客户的浏览器上，session数据放在服务器上

cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗，考虑到安全应当使用session

session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用COOKIE

## JSP和Servlet是什么关系？

**答案：**

其实这个问题在上面已经阐述过了，Servlet是一个特殊的Java程序，它运行于服务器的JVM中，能够依靠服务器的支持向浏览器提供显示内容。JSP本质上是Servlet的一种简易形式，JSP会被服务器处理成一个类似于Servlet的Java程序，可以简化页面内容的生成。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。有人说，Servlet就是在Java中写HTML，而JSP就是在HTML中写Java代码，当然这个说法是很片面且不够准确的。JSP侧重于视图，Servlet更侧重于控制逻辑。

## JSP的九大隐式对象是哪九个

**答案：**

1：request: 请求对象 在javax.servlet.ServletRequest 作用域为Request来自客服端的请求，如：FORM表单中填写的信息，常用的方法有getParameter，getParamterName 和getParamterValue通过表用获取请求对象中包含的参数值。

2:response表示客服端的响应。

3：pageContext对象为页面的上下文对象，代表当请运行页面的一些属性。

4：session：对象代码服务器与客服端所建立的会话，比如在写购物，客服轨迹跟踪，

session”是建立在cookie的基础之上的。常用方法有getId,getValues等。

5：application对象负责提供应用程序在服务端运行时的一些全局信息，方法有getMimeType等。

6：out：与response不同，通过out对象发送的内容是浏览器需要的显示内容，还可以直接想客服端编写一个有程序动态生成的HTML的文件。

7：page：page里的变量没法从index.jsp传递到test.jsp。只要页面跳转了，就不见了。

8： exception：他是一个列外的对象，当页面发生了列外，就会会创建该对象。

9：config：是在servlet初始化Servlet的时候，JSP引擎向他传递信息用的，此消息包括Servlet初始化时所需要的参数。

## 如何防止SQL注入攻击呢？

**答案：**

SQL注入：就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意）的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。

防止的方法：

1. 永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。
2. 永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。
3. 永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。
4. 不要把机密信息明文存放，请加密或者hash掉密码和敏感的信息。
5. 应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装，把异常信息存放在独立的表中。

## Mysql数据库优化

**答案：**

(1)查询时，能不用\* 就不用，尽量写全字段名。

(2)索引不是越多越好，每个表控制在6个索引以内。范围where条件的情况下，索引不起作用，比如where value<100

(3)大部分情况连接效率远大于子查询，但是有例外。当你对连接查询的效率都感到不能接受的时候可以试试用子查询，虽然大部分情况下你会更失望，但总有碰到惊喜的时候不是么...

(4)多用explain 和 profile分析查询语句

(5)有时候可以1条大的SQL可以分成几个小SQL顺序执行，分了吧，速度会快很多。

(6)每隔一段时间用alter table table\_name engine=innodb;优化表

(7)连接时注意:小表 jion 大表的原则

(8)学会用explain 和 profile判断是什么原因使你的SQL慢

(9)查看慢查询日志，找出执行时间长的SQL进行优化

(10)尽量避免使用order by

(11)因为where子句后面的条件是执行顺序是从右到左，所以尽量把能过滤掉大部分数据的条件放在最后

## Filter 的作用是什么？

**答案：**

init为初始化方法，在Filter对象被创建出来时，Servlet容器会调用该方法对filter进行初始化。

destory为销毁的方法，在过滤器对象被销毁之前，服务器会调用这个方法执行善后工作。

doFilter为过滤器中最核心的方法，对访问的请求和响应进行拦截，当过滤器拦截到对资源的访问时，服务器会自动调用该方法执行过滤代码。 我们只需要在这个方法中设计过滤器的逻辑代码即可。

## Filter的生命周期？

**答案：**

当服务器启动,web应用加载后,立即创建出这个web应用中的所有过滤器对象,创建出来后立即调用过滤器的init方法执行初始化操作.从此这些过滤器对象驻留在内存中为后续的拦截进行服务.每当拦截到资源时,都会导致dofilter方法执行.最终直到服务器关闭或web应用移除出容器时,随着web应用的销毁,过滤器对象销毁,销毁之前调用destory方法执行善后工作。

## 什么是数据库连接池及其工作原理

**答案：**

对于共享资源，有一个很著名的设计模式：资源池（resource pool）。该模式正是为了解决资源的频繁分配﹑释放所造成的问题。为解决上述问题，可以采用数据库连接池技术。数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需从“缓冲池”中取出一个，使用完毕之后再放回去。我们可以通过设定连接池最大连接数来防止系统无尽的与数据库连接。更为重要的是我们可以通过连接池的管理机制监视数据库的连接的数量﹑使用情况，为系统开发﹑测试及性能调整提供依据。

## 如何自己实现一个数据库连接池

**答案：**

思路如下：

1：利用class实现DataSource接口

2：在class的构造器一次性创建指定的链接将链接保存LinkedList中

3：实现getConnection从LinkedList返回一个链接

4：提供将链接放回方法

Public class MyDataSource inplements DataSource{

Private LinkedList<Connection> dataSource=new LinkedList<>();

Public MyDataSource(){

For( int a=0;a<1000;a++){

Try{

DriverManager.registerDriver(new SQLServerDriver());

Connection con=DriverManager.getConnection(“jdbc:sqlserver://localhost:1443;DatabaseName=liming”,”root”,”liming”

}catch(Exception e){

}

}

Public Connection getConnetion(){

Final Connection conn=dataSource.removeFirst();

}

Public void releasConnection(Connection conn){

dataSource.add(conn);

}

}

}

## http和https的区别？

**答案：**

HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了SSL（Secure Sockets Layer）协议用于对HTTP协议传输的数据进行加密，从而就诞生了HTTPS。简单来说，HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。

## Servlet的单例问题

**答案：**

Servlet是一个供其他java程序调用的类，它不能独立运行，针对客户端的多次请求，通常状况下，Servlet只会创建一个Servlet实例对象，一旦创建它就会驻留在内存中，为后续的请求提供服务，直至退出web应用为止，也就是当我们关闭了浏览器之后我们的Servlet就终止了。  
当Servlet第一次访问的时候，就被加载到内存中，以后该实例对各个请求服务，没次情况会调用一次service方法。  
这样会出现什么问题：因为Servlet是单例的，所以会出现线程安全问题

## "" 和 null的区别

**答案：**

如果说str是null，那么内存根本没创建字符串对像，并由str引用。不能调用object的方法。

如果说str是空串，那么确实存在一个由str引用的字符串对像，只不过这个字符串的值是""。长度为0；

在获取请求参数的时候为什么要这样判断呢？

if(null==str || "".equals(str)){

return "不合法参数"；

}

如果我们在表单中什么都不填 接收到的字符串就是null;

如果我们在表单中填“”，接受到的字符串是“”，但是存入数据库后，查询出来的就是null；

## Servlet的多线程同步问题

**答案：**

　　Servlet本身是单实例的，这样当有多个用户同时访问某个Servlet时，会访问该唯一的Servlet实例中的成员变量，如果对成员变量进行写入操作，那就会导致Servlet的多线程问题，即数据不一致。  
1.解决Servlet多线程同步问题的最好方式：去除实例变量，使用局部变量。  
不使用成员变量，而使用局部变量，因为局部变量在每个线程中都有各自的实例。  
　　所以对Servlet来说，如果要对某个变量做写入操作，一定不要使用成员变量，而要使用局部变量。  
2.使用同步代码块  
　　synchronized{}  
3.Servlet实现javax.serlvet.SingleThreadModel，Servlet2.4中已经废弃了该接口，此时Servlet容器将保证Servlet实例以单线程方式运行，也就是说，同一时刻，只会有一个线程执行Servlet的service()方法。

## request.getParameter()和request.getAttribute()的区别？

**答案：**

a、request.getParameter()获取的类型是String；

    request.getAttribute()获取的类型是Object

b、request.getPrameter()获取的是POST/GET传递的参数值和URL中的参数；

     request.getAttribute()获取的是对象容器中的数据值/对象

c、request.setAttribute()和request.getAttribute()可以发送、接收对象；

    request.getParamter()只能接收字符串，官方不开放request.setParamter()（也就是没有这个方法）

    setAttribute()和getAttribute()的传参原理：

    setAttribute()是应用服务器把这个对象放在该页面所对应的一块内存中去，当你的页面服务器重定向到另外一个页面时，

    应用服务器会把这块内存拷贝到另一个页面所对应的那块内存中。这个就可以通过getAttribute()获取到相应的参数值或者对象。

## JSP中动态include和静态include的区别？

**答案：**

a、静态include：语法：<%@ include file="文件名" %>，相当于复制，编辑时将对应的文件包含进来，当内容变化时，不会再一次对其编译，不易维护。

 b、动态include：语法：<jsp:include page="文件名">,能够自动检查被包含文件，当客户端对JSP文件进行请求时，会重新将对应的文件包含进来，进行实时的更新。

## 详细描述MVC。

**答案：**

基于java的web应用系统采用MVC设计模型，即用Model（模型）、View（视图）和Controller（控制）分离设计，这是目前web应用服务系统的主流设置方向。

      Model：处理业务逻辑的模块。

      View：负责页面显示，显示Model的处理结果给用户，主要实现数据到页面的转换过程。

      Controller：负责每个请求的分发，把Form数据传递给Model进行处理，处理完成后，把处理结果返回给相应的View显示给用户。

## EL表达式的功能，为什么要用EL表达式？（Expression Language）

**答案：**

功能：

a、从四个域对象中取出数据数据显示。

b、取出请求参数数据显示。

原因：

在页面中用jsp脚本和jsp表达式来获取数据显示比较麻烦

a、需要判断

b、可能需要强转

## 如何防止表单重复提交？

**答案：**

使用session技术：

a、在regist.jsp页面中生成一个为一个随机值，将其保存到session中，同时将其保存为表单的隐藏域的值。

b、在处理注册的请求时，获取session中的值，获取请求参数的值，比较两者是否相同，如果相同说明不是重复提交，请求通过同时删除session中保存的的值，如果不相同则是重复提交，不能通过。

## 什么是web容器？

**答案：**

给处于其中的应用程序组件（JSP、Servlet）提供一个环境，是JSP、Servlet直接跟容器中的变量交互，不必关注其他系统问题。

主要有web服务器来实现。例如：tomcat、weblogic、sphere、JBoss等。该容器提供的接口严格遵守J2EE规范中的web application标准。

我们把遵守以上标准的web服务器叫做J2EE的web容器。

## J2EE常用的设计模式？说明工厂模式。

**答案：**

Java中一共有23中设计模式：

Factory（工厂模式）、Builder（建造模式）、Factory Method（工厂方法模式）、ProtoType（原始模型模式）、Singleton（单例模式）、

Facade（门面模式）、Adapter（适配器模式）、Bridge（桥梁模式）、Composite（合成模式）、Decorator（装饰模式）、

FlyWeight（享元模式）、Proxy（代理模式）、Command（命令模式）、Interpreter（解释器模式）、Visitor（访问者模式）、

Iterator（迭代子模式）、Mediator（调停者模式）、Memento（备忘录模式）、Observer（观察者模式）、State（状态模式）、

Strategy（策略模式）、Template Method（模板方法模式）、Chain Of Responsibility（责任链模式）、

工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某个类的实例，通常一组类中有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

## 什么是事务？

**答案：**

事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为ACID（原子性、一致性、隔离性和持久性）属性，

只有这样才能成为一个事务：

原子性：事务必须是原子工作单元，对于其数据修改，要么全都执行，要么全都不执行。

一致性：事务在完成时，必须使所有的数据保持一致的状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如B树索引或双向链表）都必须是正确的。

隔离性：由并发事务所做的修改必须与任何其他并发事务所做的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前

的状态，要么是另一并发事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播

一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

持久性：事务完成后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

## 数据库有几种隔离级别?

**答案：**

1、Serializable (串行化)：可避免脏读、不可重复读、幻读的发生

2、Repeatable read (可重复读)：可避免脏读、不可重复读的发生。

3、Read committed (读已提交)：可避免脏读的发生。

4、Read uncommitted (读未提交)：最低级别，任何情况都无法保证

## 简述web.xml的作用

**答案：**

属于部署描述符，在整个JAVA中只要是容器都会存在部署描述符，此部署描述符可以控制整个WEB中各个组件的运行状态，也可以配置整个窗口的状态

## sql优化：（索引、范式）

**答案：**

三范式：

   第一范式（**确保每列保持原子性**）最基本范式。数据库表中所有字段值都是不可分解的原子值，就满足了第一范式。

第二范式（**确保表中的每列都和主键相关**）在第一范式上更近一层。确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关，也就是说一个表中只能保存一种数据，不可以吧多种数据保存在一张表中。

第三范式：**确保每列都和主键列直接相关，不是间接相关**。

索引：

    避免对索引字段进行计算、避免索引在字段上使用**not、<>、！=、**避免在索引上使用**IS NULL和NOT NULL**、**避免在索引列上出现数据类型转换、避免索引字段使用函数、避免建立索引的列出现空值**

## Ajax原理

**答案：**

Ajax的工作原理相当于在用户和服务器之间加了—个中间层,使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像—些数据验证和数据处理等都交给Ajax引擎自己来做, 只有确定需要从服务器读取新数据时再由Ajax引擎代为向服务器提交请求。

    Ajax其核心只有JavaScript、XMLHTTPRequest和DOM，在旧的交互方式中,由用户触发一个HTTP请求到服务器,服务器对其进行处理后再返回一个新的HTHL页到客户端, 每当服务器处理客户端提交的请求时,客户都只能空闲等待,并且哪怕只是一次很小的交互、只需从服务器端得到很简单的一个数据,都要返回一个完整的HTML页,而用户每次都要浪费时间和带宽去重新读取整个页面。而使用Ajax后用户从感觉上几乎所有的操作都会很快响应没有页面重载（白屏）的等待。

## JDBC的原理

**答案：**



## SQL注入攻击

**答案：**

SQL注入是一种将SQL代码添加到输入参数中，传递到服务器解析并执行的一种攻击手法。

SQL注入攻击是输入参数未经过滤，然后直接拼接到SQL语句当中解析，执行达到预想之外的一种行为，称之为SQL注入攻击。

## 如何防止SQL注入攻击

**答案：**

利用新对象PreparedStatement对象完成，先将SQL骨架发送给数据库服务器，然后再将参数单独发给服务器，并将参数中的关键字当做一个普通字符串来处理，进而起到防止SQL注入的问题

## 对连接池的理解

**答案：**

用来提高程序的效率，创建一个容器，容器中存放已经获取到了的数据库连接对象，对外提供获取连接和还回连接的方法，外界需要时就从容器中获取，用完就还回容器中。

## HTML和xml的区别？

**答案：**

XML是可扩展标记语言，而HTML超文本标记语言。不同之处：

1、语法有所不同。XML语法比较严谨而HTML语法比较松散。

2、用途不同。XML主要用于数据格式化存储而HTML主要用于网页的编辑。

## 在JS中==和===的区别？

**答案：**

简单来说： == 代表相同， ===代表严格相同, 为啥这么说呢，

这么理解： 当进行双等号比较时候： 先检查两个操作数数据类型，如果相同， 则进行===比较， 如果不同， 则愿意为你进行一次类型转换， 转换成相同类型后再进行比较， 而===比较时， 如果类型不同，直接就是false.

## SQL优化

**答案：**

1. SELECT子句中避免使用‘\*’
2. SQL语句用大写的
3. 用IN来替换OR
4. 查询语句中不要使用 \*
5. 尽量减少子查询，使用关联查询（left join,right join,inner join）替代
6. 减少使用IN或者NOT IN ,使用exists，not exists或者关联查询语句替代
7. or 的查询尽量用 union或者union all 代替
8. 合理的增加冗余的字段（减少表的联接查询）
9. 增加中间表进行优化（这个主要是在统计报表的场景，

## Tomcat配置,部署优化

**答案：**

1. 内存优化:Tomcat依赖于JVM,可以配置JVM的内存配置

2. 最大连接数配置(并发能力)

通常搭配Nginx提升Tomcat的并发性能

## 自动刷新,定时刷新

**答案：**

自动刷新不仅可以实现一段时间之后自动跳转到另一个页面，还可以实现一段时间之后自动刷新本页面。Servlet中通过HttpServletResponse对象设置Header属性实现自动刷新例如：

Response.setHeader("Refresh","1000;URL=http://localhost:8080/servlet/example.htm");

其中1000为时间，单位为毫秒。URL指定就是要跳转的页面（如果设置自己的路径，就会实现没过一秒自动刷新本页面一次）

## BS和CS的区别？

**答案:**

1. C/S用户固定，并且处于相同区域，要求拥有相同的[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)。B/S要有[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)和浏览器就行。与[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank)平台无关。
2. C/S客户端的计算机电脑配置要求较高。B/S客户端的计算机电脑配置要求较低。
3. C/S每一个客户端都必须安装和配置软件,客户端不必安装，使用浏览器访问，易推广。B/S最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件。
4. C/S每一个客户端都要升级程序。可以采用自动升级。BS客户端不必安装及维护。
5. C/S一般面向相对固定的用户群，程序更加注重流程，它可以对权限进行多层次校验，提供了更安全的存取模式，对信息安全的控制能力很强。一般高度机密的信息系统采用C/S结构适宜。

## 事务有哪些特性

事务的特性:ACID

A - 原子性 Atomic

数据操作的最小单元是事务，而不是SQL语句

C - 一致性 Consistency

转账前 a+b = 100

转帐后 a+b = 100

I - 隔离性 Isolation

一个事物进行中时，另一事物不能操作数据

D - 持久性 Durancy

事务没有提交之前，数据操作只保存在日志文件中

提交事务之后，数据持久生效

## html和jsp的区别及优缺点

**答案:**

 HTML（Hypertext Markup Language）文本标记语言，它是静态页面，和JavaScript一样解释性语言，为什么说是解释性语言呢？因为，只要你有一个浏览器那么它就可以正常显示出来，而不需要指定的编译工具，只需在TXT文档中写上HTML标记就OK。  
JSP（Java Server Page）看这个意思就知道是Java服务端的页面，所以它是动态的，它是需要经过JDK编译后把内容发给客户端去显示，我们都知道，Java文件编译后会产生一个class文件，最终执行的就是这个class文件，JSP也一样，它也要编译成class文件！JSP不止要编译，它还得要转译，首先把JSP转译成一个Servlet文件，然后在编译成class文件。当用户访问JSP时就执行了class文件，最 终......

1.最简单的区别就是，HTML能直接打开，jsp只能发布到Tomact等服务器上才能打开

2.定义上HTML页面是静态页面可以直接运行，JSP页面是动态页它运行时需要转换成servlet

3.他们的表头不同，这个是JSP的头“ <%@ page language="java" import="java.util.\*" pageEncoding="gbk"%>”在表头中有编码格式和导入包等

4.也是很好区分的在jsp中用<%%>就可以写Java代码了，而html没有<%%>

## HTML和Servlet的异同

**答案:**

不同： Html是静态，servlet是动态 html页面由服务器直接返回， servlet是用来处理客户请求，并返回html页面 ,Servlet需要服务器调用servlet方法生成动态html页面，且需要在web.xml中配置url路径

## request ，response，session 和 application是怎么用的

**答案:**

1、Request是客户端向服务端发送请求

2、Response是服务端对客户端请求做出响应

3、Session在servlet中不能直接使用，需要通过getSession()创建，如果没有设定它的生命周期，或者通过invildate()方法销毁，关闭浏览器session就会消失

4、Application不能直接创建，存在于服务器的内存中，由服务器创建和销毁

## Request和Session的取值区别，以及出现乱码的解决方式（不能在java代码中设置）

**答案:**

1、Request可以通过getAttribute()方法直接取值，也可通过getParameter()取值

2、Session需要通过request.getSession().getAttribute()才能取值

3、Request是针对一次请求，Session是针对整个会话

4、在页面通过contentType，pageEncoding，content设置编码格式，必须要一致

## 说明一下jsp中<jsp: include page..>和<%@ include file%>的区别

**答案:**

动态导入

是行为元素、是在请求处理阶段引入的，引入执行页面或servlet所生成的应答文本

先编译，后包含,就是将每个jsp页面都单独转化成html页面，最后再将所有的html页面相加，如果有相同变量不会冲突

<%@ include file="" %> 静态导入

是指令元素，是编译时包含，引入静态文本(html,jsp),在JSP页面被转化成servlet之前和它融和到一起。先包含，后编译

## Spring 在ssm中起什么作用？

**答案:**

Spring：轻量级框架

作用：Bean工厂，用来管理Bean的生命周期和框架集成。

两大核心：

1、IOC/DI(控制反转/依赖注入) ：把dao依赖注入到service层，service层反转给action层，Spring顶层容器为BeanFactory。

2、AOP：面向切面编程

## Spring的配置文件中的内容？

**答案:**

开启事务注解驱动

事务管理器

开启注解功能，并配置扫描包

配置数据库

配置SQL会话工厂，别名，映射文件

不用编写Dao层的实现类

## Spring主要使用了什么模式？

**答案:**

工厂模式：每个Bean的创建通过方法

单例模式：默认的每个Bean的作用域都是单例

代理模式：关于Aop的实现通过代理模式

## Struts2 和 SpringMvc的区别?

**答案:**

入口不同：

Struts2：filter过滤器

SpringMvc：一个Servlet即前端控制器

开发方式不同：

Struts2：基于类开发，传递参数通过类的属性，只能设置为多例

SpringMvc：基于方法开发(一个url对应一个方法)，请求参数传递到方法形参，可以为单例也可以为多例(建议单例)

请求方式不同：

Struts2：值栈村塾请求和响应的数据，通过OGNL存取数据

SpringMvc：通过参数解析器将request请求内容解析，给方法形参赋值，将数据和视图封装成ModelAndView对象，最后又将ModelAndView中的模型数据通过request域传输到页面，jsp视图解析器默认使用的是jstl。

## Mybatis的好处？

**答案:**

把Sql语句从Java中独立出来。

封装了底层的JDBC，API的调用，并且能够将结果集自动转换成JavaBean对象，简化了Java数据库编程的重复工作。

自己编写Sql语句，更加的灵活。

入参无需用对象封装（或者map封装）,使用@Param注解