# Servlet

## 面试题

### Servlet是什么？

Servlet是sun公司提供的一门用于开发动态web资源的技术。

按照这套规范写出来的Servlet可以放置到web应用中在Servlet容器中运行。

### 说明Servlet接口中方法及方法的作用？

init(ServletConfig config)

初始化方法，会在Servlet被创建出来后立即执行，做一些初始化的工作

destroy()

销毁方法，会在Servlet被销毁之前执行，做一些善后工作

service(ServletRequest req, ServletResponse res)

服务方法，每当有请求访问Servlet时，此方法执行，处理请求的代码要写到这个方法里。

### 描述Servlet调用过程？

（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？

（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。

# HTTP

## 面试题

### 什么是http协议？

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？

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 ?

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。

### 请求方式有几种？

一共有7种请求方式:GET、POST、HEAD、OPTIONS、PUT、DELETE和TARCE

HTTP/1.1协议中共定义了八种方法（有时也叫“动作”）来表明Request-URI指定的资源的不同操作方式：

1. GET 向特定的资源发出请求。注意：GET方法不应当被用于产生“副作用”的操作中，例如在web app.中。其中一个原因是GET可能会被网络蜘蛛等随意访问。
2. POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
3. HEAD 向服务器索要与GET请求相一致的响应，只不过响应体将不会被返回。这一方法可以在不必传输整个响应内容的情况下，就可以获取包含在响应消息头中的元信息。
4. OPTIONS 返回服务器针对特定资源所支持的HTTP请求方法。也可以利用向Web服务器发送'\*'的请求来测试服务器的功能性。
5. PUT 向指定资源位置上传其最新内容。
6. DELETE 请求服务器删除Request-URI所标识的资源。
7. TRACE 回显服务器收到的请求，主要用于测试或诊断。
8. 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。

### get和post请求的区别？

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()。

### 列出常见的请求头及其功能？

Accept-Charset: ISO-8859-1

-- 浏览器可以接受的字符集编码

Accept-Encoding: gzip,compress

-- 浏览器可以接受的压缩格式

Referer: http://www.it315.org/index.jsp

-- 这是和防盗链相关的头,对当前资源的访问来自哪个页面的超链接

HTTP响应结构

一个典型的HTTP响应分为 一个状态行 若干响应头 一个空行 实体内容。

### 状态码及表示的含义？

200 表示成功处理完成请求

302 表示重定向

304 表示通知浏览器使用缓存中的资源

307 表示通知浏览器使用缓存中的资源

404 找不到资源

500 服务器端错误

### 列出常见的响应头及其功能？

Content-Type: text/html; charset=GB2312

-- 服务器发送的数据是什么格式的,如果是字符格式的数据,则还可以通知服务器发送的数据使用的是什么编码,浏览器会根据这个头指定的编码决定使用哪个编码来打开收到的数据

Refresh: 1;url=http://www.it315.org

-- 定时刷新相关的头,通知浏览器,过几秒后自动刷新访问哪个地址

Content-Disposition: attachment;filename=aaa.zip

-- 通知浏览器以附件的形式打开发送过去的数据,是和文件下载相关的头

Expires: -1

-- 通知浏览器是否缓存

Cache-Control: no-cache

-- 通知浏览器是否缓存

Pragma: no-cache

-- 通知浏览器是否缓存

# Request

## 面试题

### request对象能获取哪些客户机相关的信息？

getRequestURL方法返回客户端发出请求完整URL

getRequestURI方法返回请求行中的资源名部分

getQueryString 方法返回请求行中的参数部分

getRemoteAddr方法返回发出请求的客户机的IP地址

getMethod 得到客户机请求方式

getContextPath 获得当前web应用虚拟目录名称

### 请求乱码产生的原因？

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?

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

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。

# ServletConfig

## 面试题

### ServletConfig是什么？

ServletConfig代表当前Servlet在web.xml中的配置信息。从一个servlet被实例化后，对任何客户端在任何时候访问有效，但仅对本servlet有效，一个servlet的ServletConfig对象不能被另一个servlet访问。

# Response

## 面试题

### Response向外输出数据有哪些方法？

有两种方法：

PrintWriter getWriter()

ServletOutputStream getOutputStream();

其中getWriter获取的是字符流，可以输出字符数据到客户端。

getOutputStream获取的是字节流，可以输出字节数据到客户端。

### 如何处理响应乱码？

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。

# ServletContext

## 面试题

### 简述ServletContext生命周期？

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。

### ServletContext的作用范围是什么？

**ServletContext,是一个全局的储存信息的空间，服务器开始，其就存在，服务器关闭，其才释放。**

# 实现资源跳转

## 面试题

### 转发与重定向的比较？

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

## 面试题

### Session生命周期？

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？

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

## 面试题

### cookie与session的区别

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

### 描述Cookie个数限制及大小？

各浏览器之间对cookie的限制不同，

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | IE6.0 | IE7.0/8.0 | Opera | FF | Safari | Chrome |
| cookie个数 | 每个域为20个 | 每个域为50个 | 每个域为30个 | 每个域为50个 | 没有个数限制 | 每个域为53个 |
| cookie大小 | 4095个字节 | 4095个字节 | 4096个字节 | 4097个字节 | 4097个字节 | 4097个字节 |

在进行页面cookie操作的时候，应该尽量保证cookie个数小于20个，总大小 小于4KB。

# JSP

## 面试题

### JSP和Servlet是什么关系？

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，在MVC架构模式中，JSP适合充当视图（view）而Servlet适合充当控制器（controller）。

### include指令与include动作的区别是什么？

从字面意思看，两者都有包含的意思，并且从运行的结果上来看，两者的功能也非常类似，但是两者的运行过程相差很大。

<%@ include%>指令是编译时语法，也就是在编译的时候，把指令所指向的目标文件的内容拷贝到指令所在的位置，替换指令，最终形成一个文件，在运行的时候只有一个文件。

<jsp:include>动作是运行时语法，包含<jsp:include>动作的文件在执行到这个标签的时候，会转向执行标签所指向的目标文件，执行完目标文件之后，再接着执行标签后的内容，在运行的时候，涉及到两个文件，就像方法调用一样。

那么什么时候应该使用<%@ include%>指令，什么时候使用<jsp:include>标签呢？因为<%@ include%>指令是静态的，而<jsp:include>是动态的，所以如果某一段代码肯定会执行，则可以使用<%@ include%>指令，如果某一段代码有可能执行，有可能不执行，需要根据运行时候的状态，这时候可以使用<jsp:include>。

在实际应用中，很多网站的导航部分和版权信息部分都是相同的，在每个文件中都会出现，所以可以考虑把这些内容放在单独的文件中，然后使用<%@ include%>指令引用使用。

### JSP有哪些内置对象？作用分别是什么？

JSP有9个内置对象：

- request：封装客户端的请求，其中包含来自GET或POST请求的参数；

- response：封装服务器对客户端的响应；

- pageContext：通过该对象可以获取其他对象；

- session：封装用户会话的对象；

- application：封装服务器运行环境的对象；

- out：输出服务器响应的输出流对象；

- config：Web应用的配置对象；

- page：JSP页面本身（相当于Java程序中的this）；

- exception：封装页H 面抛出异常的对象。

### 讲解JSP中的四种作用域。

JSP中的四种作用域包括page、request、session和application，具体来说：

- page代表与一个页面相关的对象和属性。

- request代表与Web客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个Web组件；需要在页面显示的临时数据可以置于此作用域。

- session代表与某个用户与服务器建立的一次会话相关的对象和属性。跟某个用户相关的数据应该放在用户自己的session中。

- application代表与整个Web应用程序相关的对象和属性，它实质上是跨越整个Web应用程序，包括多个页面、请求和会话的一个全局作用域。

### 在四大域中搜寻属性，搜寻的顺序是什么？

page域、request域、session域、application域从小域到大域开始搜索，如果搜索到就直接获取该值，如果所有域中都找不到，返回一个null。

### EL表达式中有哪些隐含对象可用？

pageContext、pageScope、requestScope、sessionScope、applicationScope、param、paramValues、header、headerValues、cookie、initParam

# MySQL

## 面试题

### 如何开启、停止 MySQL 服务？

运行命令service mysqld start开启服务；运行命令service mysqld stop停止服务

### Innodb与MyISAM在事务处理上有什么区别？

MyISAM类型的表强调的是性能，其执行数度比InnoDB类型更快，但是不提供事务支持，InnoDB提供事务支持事务。

原来是MyISAM类型不支持事务处理等高级处理，而InnoDB类型支持。 MyISAM类型的表强调的是性能，其执行数度比InnoDB类型更快，但是不提供事务支持，而InnoDB提供事务支持已经外部键等高级数据库功能。综述，就可以根据数据表不同的用处是用不同的存储类型。而且MyISAM是文件存储的，可以进行直接在不同操作系统间拷贝使用。

InnoDB 给 MySQL 提供了具有事务(commit)、回滚(rollback)和崩溃修复能力(crash recovery capabilities)的事务安全(transaction-safe (ACID compliant))型表。InnoDB 提供了行锁(locking on row level)，提供与 Oracle 类型一致的不加锁读取(non-locking read in SELECTs)。这些特性均提高了多用户并发操作的性能表现。在InnoDB表中不需要扩大锁定(lock escalation)，因为 InnoDB 的列锁定(row level locks)适宜非常小的空间。InnoDB 是 MySQL 上第一个提供外键约束(FOREIGN KEY constraints)的表引擎。InnoDB 的设计目标是处理大容量数据库系统，它的 CPU 利用率是其它基于磁盘的关系数据库引擎所不能比的。在技术上，InnoDB 是一套放在 MySQL 后台的完整数据库系统，InnoDB 在主内存中建立其专用的缓冲池用于高速缓冲数据和索引。 InnoDB 把数据和索引存放在表空间里，可能包含多个文件，这与其它的不一样，举例来说，在 MyISAM 中，表被存放在单独的文件中。InnoDB 表的大小只受限于操作系统的文件大小，一般为 2 GB。InnoDB所有的表都保存在同一个数据文件 ibdata1 中（也可能是多个文件，或者是独立的表空间文件），相对来说比较不好备份，可以拷贝文件或用navicat for mysql。

MyISAM，每张MyISAM 表被存放在三个文件 ：frm 文件存放表格定义。 数据文件是MYD (MYData) 。 索引文件是MYI (MYIndex) 引伸。

因为MyISAM相对简单所以在效率上要优于InnoDB，小型应用使用MyISAM是不错的选择。

MyISAM表是保存成文件的形式,在跨平台的数据转移中使用MyISAM存储会省去不少的麻烦

### 如何防止SQL注入攻击呢？

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。

### 创建索引的好处

对于查询占主要的应用来说，索引显得尤为重要。很多时候性能问题很简单的就是因为我们忘了添加索引而造成的，或者说没有添加更为有效的索引导致。如果不加索引的话，那么查找任何哪怕只是一条特定的数据都会进行一次全表扫描，如果一张表的数据量很大而符合条件的结果又很少，那么不加索引会引起致命的性能下降。但是也不是什么情况都非得建索引不可，比如性别可能就只有两个值，建索引不仅没什么优势，还会影响到更新速度，这被称为过度索引。

### 索引列上计算引起的索引失效及优化措施以及注意事项

创建索引、优化查询以便达到更好的查询优化效果。但实际上，MySQL有时并不按我们设计的那样执行查询。MySQL是根据统计信息来生成执行计划的，这就涉及索引及索引的刷选率，表数据量，还有一些额外的因素。

Each table index is queried, and the best index is used unless the optimizer believes that it is more efficient to use a table scan. At one time, a scan was used based on whether the best index spanned more than 30% of the table, but a fixed percentage no longer determines the choice between using an index or a scan. The optimizer now is more complex and bases its estimate on additional factors such as table size, number of rows, and I/O block size.

简而言之，当MYSQL认为符合条件的记录在30%以上，它就不会再使用索引，因为mysql认为走索引的代价比不用索引代价大，所以优化器选择了自己认为代价最小的方式。事实也的确如此

是MYSQL认为记录是30%以上，而不是实际MYSQL去查完再决定的。都查完了，还用什么索引啊？！

MYSQL会先估算，然后决定是否使用索引。

### Mysql数据库优化

(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

## 面试题

### Filter 的作用是什么？

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？

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。

# 其他

## 面试题

### web.xml 的作用？

用于配置Web应用的相关信息，如：

监听器：

<listener> <listener-class>com.itheima.listener.MyServletContextListener</listener-class>

</listener>

<listener> <listener-class>com.itheima.listener.MyHttpSessionListener</listener-class>

</listener>

设置全栈编码：

<context-param>

<description>全站编码设置</description>

<param-name>encode</param-name>

<param-value>utf-8</param-value>

</context-param>

设置统一错误页面

<error-page>

<error-code>500</error-code>

<location>/error/500.jsp</location>

</error-page>

<error-page>

<error-code>404</error-code>

<location>/error/404.jsp</location>

</error-page>

### 使用标签库有什么好处？

使用标签库的好处包括以下几个方面：

分离JSP页面的内容和逻辑，简化了Web开发；

开发者可以创建自定义标签来封装业务逻辑和显示逻辑；

标签具有很好的可移植性、可维护性和可重用性；

避免了对Scriptlet（小脚本）的使用（很多公司的项目开发都不允许在JSP中书写小脚本）

### 如何自定义JSP标签？

自定义JSP标签包括以下几个步骤：

编写一个Java类实现实现Tag/BodyTag/IterationTag接口(通常不直接实现这些接口而是继承TagSupport/BodyTagSupport/SimpleTagSupport类)

重写doStartTag()、doEndTag()等方法，定义标签要完成的功能

编写扩展名为tld的标签描述文件对自定义标签进行部署，tld文件通常放在WEB-INF文件夹或其子目录

在JSP页面中使用taglib指令引用该标签库