Mysql

**SQL优化**

* 外连接代替子查询
* 使用JOIN时候，应该用小表驱动大表，（left join 左边表结果尽量小，如果有条件应该放到左边先处理，right join同理反向）
* 用绑定变量代替传入sql语句的值，这样类似查询可以重用这个执行计划，减少数据库解析语句的负担
* con.prepareStatement("UPDATE employees SET salay = ? WHERE id = ?");
* pstmt.setBigDecimal(1, 150.00);
* pstmt.setInt(2, id);
* 建立组合索引时把最常用的字段放在最左端

**什么是存储过程？用什么来调用？**

存储过程是一段预编译的SQL语句，只需创建一次，以后可以多次调用

参数有三种模式：IN，OUT或INOUT

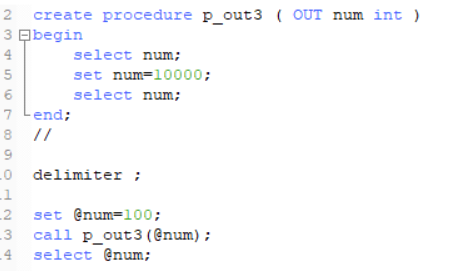
OUT

被声明的值可在存储过程内部被改变，并返回给原来的值

不接受外部传入的数据(得不到传进来的值)

IN

在存储过程中更改了IN参数的值，在存储过程结束后仍保留其原始值@num原来的值不变



**引擎**

MYISAM misen：默认采用

1. 不支持事务/外键，使用表锁（读锁（共享）和写锁（独占）
2. MyISAM读写互相阻塞：不仅会在写入的时候阻塞读取，MyISAM还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读, InnoDB 读写阻塞与事务隔离级别相关
3. 支持全文索引

表锁：它会锁定整张表，一个用户在对表进行写操作前需要先获得写锁，这将会阻塞其他用户对该表的所有读写操作

INNODB：

1. 支持事务/外键, 有4种事务隔离级别，适合需要执行大量的INSERT或UPDATE操作，但查询速度不如Myisam
2. 支持行锁（共享锁S和排他锁X）高并发性能更好

**讲讲MySQL事务必须满足4个条件（ACID）：**

* **原子性：事务是最小工作单元，一个事务（transaction）中的所有操作，要么全部完成，要么全部失败回滚**
* **一致性：事务开始之前和结束以后，数据库保持完整性约束**
* **隔离性：事务的修改在提交前，其他事务是不可见**
* **持久性：事务对数据的修改永久保存在数据库中**

Redis

**redis提供了两种持久化的方式，分别是RDB（Redis DataBase）和AOF（Append Only File）**

RDB，每隔一段时间，将redis存储的数据生成快照(数据库在某一时刻的数据信息，相当于一个只读数据库 备份恢复和读写分离)并存储到磁盘等介质上；

redis在进行数据持久化的过程中，会先将数据写入到一个临时文件中，待持久化过程都结束了，才会用这个临时文件替换上次持久化好的文件

AOF，将redis执行过的所有写指令(push/pop~)记录下来，在下次redis重新启动时，只要把这些写指令从前到后再重复执行一遍，就可以实现数据恢复了。

RDB和AOF两种方式也可以同时使用，在这种情况下，如果redis重启的话，则会优先采用AOF方式来进行数据恢复，这是因为AOF方式的数据恢复完整度更高。RDB时间间隔内的数据可能没有保存到。

如果没有数据持久化的需求，也完全可以关闭RDB和AOF方式，这样的话，redis将变成一个纯内存数据库，就像memcache一样

**redis 常见数据结构以及使用场景//用到redis哪些方面（除了缓存）**

**String** 常用命令: set,get,decr-,incr+

String数据结构是key-value类型，value其实不仅可以是String，也可以是数字

常规计数：粉丝数

**Hash**

常用命令： hget,hset,hgetall 等

hash 是一个 string 类型的 field 和 value 的映射表

hash 特别适合用于存储对象

可以存储用户信息，商品信息

**List**

常用命令: lpush,lpop, lrange等

双向链表，即可以支持双向向查找和遍历

比如用户注列表，lrng分页查询等

**Set**

常用命令： sadd,spop,smembers

set 与list类似是一个列表的功能，但 set 是可以自动排重

有求交集、并集、差集的操作

可以实现如共同关注、点赞等功能。这个过程也就是求交集的过程

**Sorted Set**

常用命令： zadd ,zrem移除,zcard元素数量等

和set相比，sorted set增加了一个参数score，使得集合中的元素能够按score进行有序排列

举例： 直播系统实时礼物排行榜

**Servlet 是什么？**

Java Servlet 是一个java类，运行在 Web 服务器,用于生成动态的内容

**Servlet 的生命周期？**

**1. 加载 Servlet**：当服务器 第一次访问 Servlet 的时候，Tomcat 会负责创建 Servlet 的实例。

**2. 初始化 Servlet**：服务器会调用Servlet的 init() 方法来初始化这个对象。

**3. 处理服务**：当浏览器访问 Servlet 的时候，Servlet 会调用 service() 方法处理请求。

**4. 销毁**：当 Tomcat 关闭或者要删除Servlet的时候，会自动调用 destroy() 方法，让该实例所占用的资源释放掉。一个 Servlet 如果长时间不被使用的话，也会被 Tomcat 自动销毁

**Servlet 是单例的吗？为什么？**

Servlet 是单例的，**服务器只创建一个Servlet对象**，驻留在内存中，**为后续的请求做服务，直到服务器关闭**

### GET 和 POST 的区别

GET

http://localhost/test.html?name=wmyskxz&sex=male

缺点：

暴露请求信息，**不安全**  
只允许 ASCII a s key字符，可传输的信息有限，不能上传图片

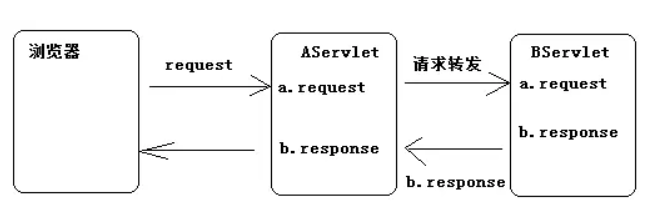
POST

隐藏了请求信息，**较安全**

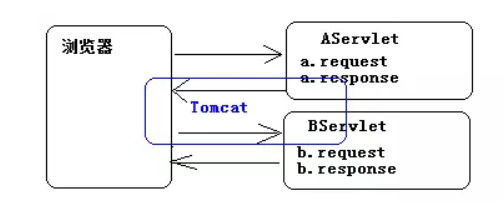
数据类型没有限制，允许二进制数据，可以做图片的上传

### forward 与 redirect 的区别

### 转发（forward）



**重定向（redirect）**



1. 数据共享

forward:转发页面和转发到的页面可以共享request里面的数据

redirect:不能共享数据

1. 地址栏显示

forward是服务器请求资源, 服务器直接访问目标地址的URL,把那个URL的响应内容读取过来, 然后把这些内容再发给浏览器. 浏览器不知道服务器发送的内容从哪里来的, 所以它的地址栏还是原来的地址

redirect是服务端接收到请求后，返回一个状态码, 告诉浏览器重新去请求那个地址，所以地址栏显示的是新的URL

web服务器接受后发送302状态码响应及对应新的location给客户浏览器--->客户浏览器发现是302响应，则自动再发送一个新的http请求，请求url是新的location地址

500服务器内部错误 300 重定向 400请求错误

1. 请求数：

Forward浏览器请求一次，

redirect两次：服务器端在响应第一次请求的时候，让浏览器再向另外一个URL发出请求，从而达到转发的目的

1. WEB-INF资源

**f**orward【可**以】访问 WEB-INF 中的资源**

redirect【不可以】访问 WEB-INF 中的资源

Tomcat

* bin:存放启动和关闭Tomcat的脚本文件
* conf:存放Tomcat服务器的配置文件
* lib:存放Tomcat服务器的所需的jar包
* logs:存放Tomcat的日志文件
* webapps:存放web资源目录

**用Tomcat的哪些方面**

Context去掉包路径

**Tomcat 有哪几种Connector 运行模式(优化)？**

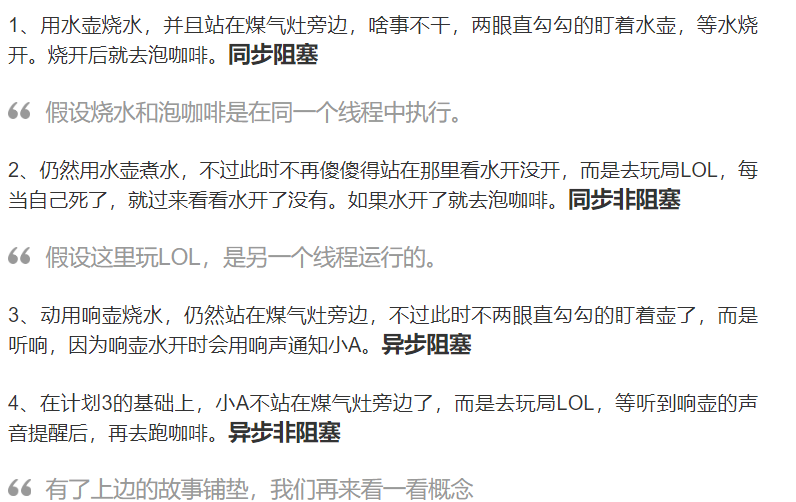
protocol：协议类型，可选类型有四种，分别为BIO，NIO，NIO2和APR

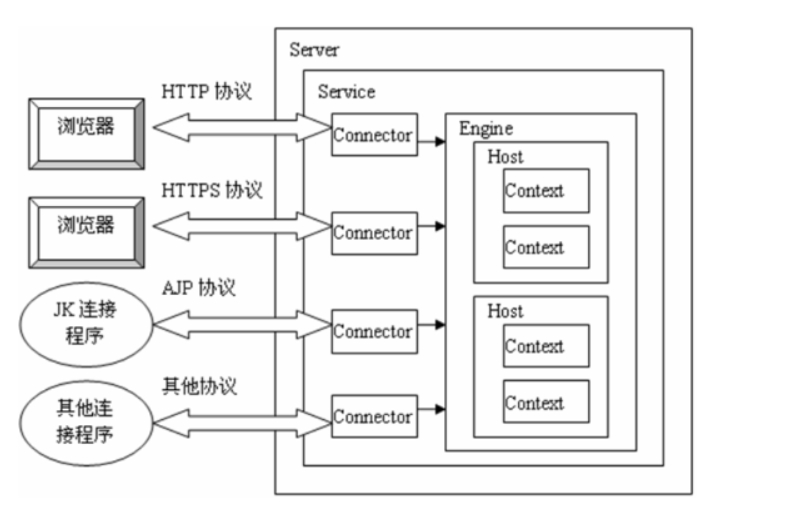
* bio: 传统的Java I/O操作，同步且阻塞，适合简单项目 传统的Java I/O操作
* nio: 同步非阻塞IO ，适合后台需要耗时完成请求的操作

java.nio包及其子包

Java nio是一个基于缓冲区、并能提供非阻塞I/O操作的Java API

* nio.2: 异步非阻塞IO





Tomcat服务器的启动是基于一个server.xml文件的，Tomcat启动的时候首先会启动一个Server，Server里面就会启动Service，Service里面就会启动多个"Connector(连接器)"，每一个连接器都在等待客户机的连接，当访问服务器上面的web资源时，首先是连接到Connector(连接器)，Connector(连接器)接收请求并将接收的请求封装为Request和Response，再将请求交给一个Engine(引擎)去处理，Engine(引擎)接收到请求后就会解析用户想要访问的Host，然后将请求交给相应的Host，Host收到请求后就会解析出用户想要访问这个Host下面的哪一个Web应用(即Context)

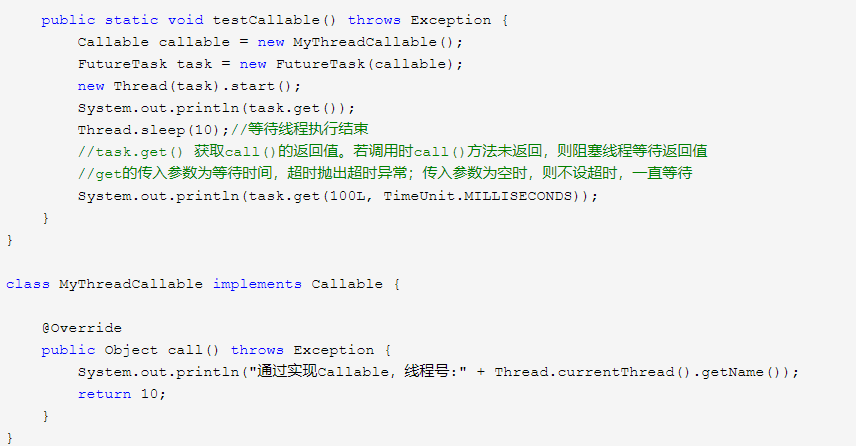
Connector监听端口 localhost在Host节点配置

多个Service 一个Service中一个engine

线程

**三种创建线程的方法：**

* **通过实现 Runnable 接口；**
* **通过继承 Thread 类本身；**
* **通过 Callable 接口和 Future接口 创建线程**



**Thread类的sleep()方法和对象的wait()方法都可以让线程暂停执行，它们有什么区别?**

* sleep()方法是线程类（Thread）的静态方法，调用此方法会让当前线程暂停执行指定的时间，将执行机会（CPU）让给其他线程，但是对象的锁依然保持，因此休眠时间结束后会继续执行
* 调用对象的wait()方法导致当前线程放弃对象的锁（线程暂停执行），进入对象的等待池（wait pool），只有调用对象的notify()方法（或notifyAll()方法）时才能唤醒等待池中的线程进入等锁池（lock pool），如果线程重新获得对象的锁就可以进入就绪状态

join方法，等待调用该方法的线程执行完毕后 再往下继续执

**线程的sleep()方法和yield()**you de**方法有什么区别？**

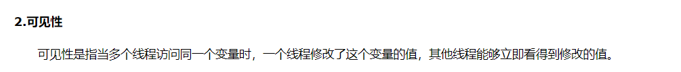
* sleep()方法给其他线程运行机会时不考虑线程的优先级；yield()方法只会给相同优先级或更高优先级的线程运行的机会；
* 线程执行sleep()方法后转入阻塞（blocked）状态，而执行yield()方法后转入就绪状态；

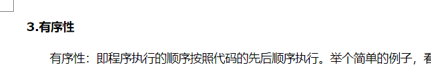
当一个线程使用了这个方法之后，它就会把自己CPU执行的时间让掉，使当前线程从执行状态（运行状态）变为可执行态（就绪状态），让自己或者其它的线程运行。cpu调度器会从众多的可执行态里选择，也就是说，当前也就是刚刚的那个线程还是有可能会被再次执行到的，并不是说一定会执行其他线程而该线程在下一次中不会执行到了

* sleep()方法声明抛出InterruptedException(中断异常)，而yield()方法没有声明任何异常；

并发特性

原子性是指在一个操作中就是cpu不可以在中途暂停然后再调度，既不被中断操作，要不执行完成，要不就不执行





synchronized可作用于一段代码或方法，既可以保证可见性，原子性

**Lock 同，但要求手工释放锁和具有公平性**

Volatile va le chou 保证可见性和禁止指令重排 不能保证原子性

四种作用域

Page/request

session:用于存储用户登录信息/购物车

application:用于存在多个客户端在当前应用中共享的数据，典型：在线人数统计、初始化全局资源、得到项目配置信息

Vue和JQ区别

Vue对数据进行操作不再需要引用相应的DOM对象;

View(视图) 和 Model(数据) 通过ViewModel双向数据绑定，其中一个变化另一个也会变

Vue具有组件化：删除某个组件不会影响其他组件

Http和TCP

Http协议是建立在TCP协议基础之上的，当浏览器发出一次Http请求。Http会通过TCP建立起一个到服务器的连接通道，当本次请求需要的数据完毕后，Http会立即将TCP连接断开，这个过程是很短的。所以Http连接是一种短连接，是一种无状态的连接

HTTP1.1规定了默认保持长连接（HTTP persistent connection ，也有翻译为持久连接），数据传输完成了保持TCP连接不断开（不三次握手消耗），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

Connection: Keep-alive/Close

TCP的keep alive是检查当前TCP连接是否活着；HTTP的Keep-alive是要让一个TCP连接活久点

Http无状态，前后请求没有关联，可以通过Cookie和Session区别用户