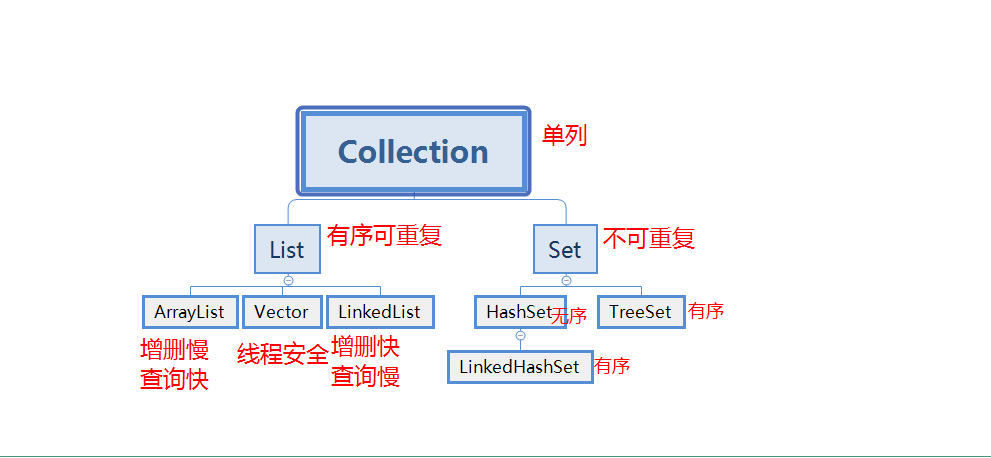
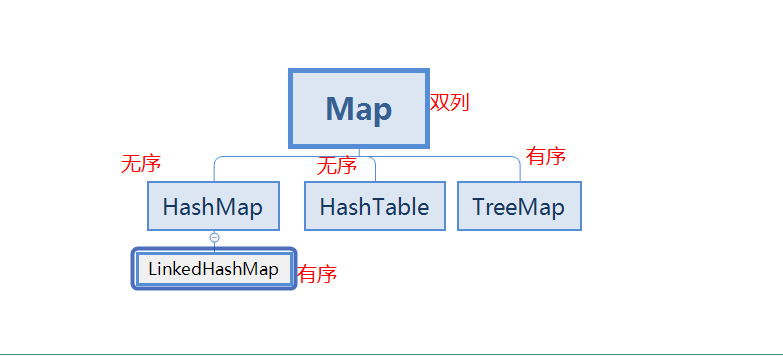
# 第一部分：基础知识

## 1. 常用的集合有哪些，谈谈对它们的理解?





**Collection接口：**单列集合，有两个子接口

1.**List接口（有序可重复）**：

有三个实现类

LinkedList：基于链表实现，每一个元素存储本身内存地址的同时还存储下一个元素的地址。链表增删快，查找慢；

ArrayList：基于数组；每次增删都要创建新的数组，但数组有索引。数组增删慢，查找快

Vector：基于数组，线程安全的，效率低

2.**Set接口（不可重复）：**

有两个实现类

HashSet:存储的元素无序，不可重复，底层是哈希表（无序）

TeeSet；存储单元有序，不可重复，

LinkedHashSet:存储的元素有序，不可重复，底层是哈希表和链表的结合

**Map接口：**双列集合

有三个实现类（HashMap，HashTable，TreeMap）

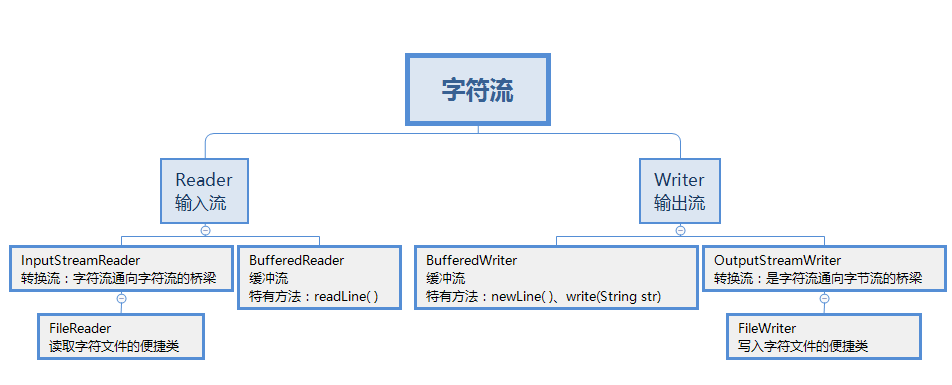
HashMap：非线程安全，高效，支持null；（无序）

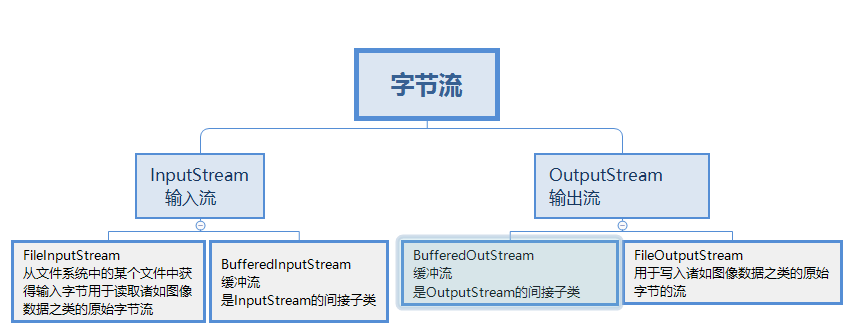
LinkedHashMap：是HashMap的一个子类，保存了记录的插入顺序（有序）

HashTable：线程安全，低效，不支持null；（无序）

TreeMap: 能够把它保存的记录根据键排序，默认是键值的升序排序（有序）

## 2. IO流





**字节流和字符流的区别**

1.字节流读取的时候，读到一个字节就返回一个字节； 字符流使用了字节流读到一个或多个字节（中文对应的字节数是两个，在UTF-8码表中是3个字节）时。先去查指定的编码表，将查到的字符返回。   
 2.字节流可以处理所有类型数据，如：图片，MP3，AVI视频文件，而字符流只能处理字符数据。只要是处理纯文本数据，就要优先考虑使用字符流，除此之外都用字节流。

**一些特别的流类型**

**转换流**，转换流只有字节流转换为字符流，因为字符流使用起来更方便，我们只会向更方便使用的方向转化。如：InputStreamReader与OutputStreamWriter。

**缓冲流**，有关键字Buffered，也是一种处理流，为其包装的流增加了缓存功能，提高了输入输出的效率，增加缓冲功能后需要使用flush()才能将缓冲区中内容写入到实际的物理节点。但是，在现在版本的Java中，只需记得关闭输出流（调用close()方法），就会自动执行输出流的flush()方法，可以保证将缓冲区中内容写入。

**对象流**，有关键字Object，主要用于将目标对象保存到磁盘中或允许在网络中直接传输对象时使用（对象序列化）。

**推回输入流**，有关键字PushBack，当程序调用推回输入流的unread()方法时，系统回把指定数组内容的内容推回到一个推回缓冲区中，在调用read()方法读入内容时，就先从推回缓冲区中读取，直到读完推回缓冲区中内容后才会从原输入流中读取。

## 3. 线程和进程的区别？

**进程：**具有一定独立功能的程序关于某个数据集合上的一次运行活动，是操作系统进行资源分配和调度的一个独立单位

**线程：**是进程的一个实体，是cpu调度和分派的基本单位，是比进程更小的可以独立运行的基本单位

**特点：**线程的划分尺度小于进程，这使多线程程序拥有高并发性

进程在运行时各自内存单元相互独立，线程之间内存共享，这使多线程编程可以拥有更好的性能和用户体验

注意：多线程编程对于其它程序是不友好的，占据大量cpu资源。

## 4. 启动一个线程是调用run()方法还是start()方法？

启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM 调度并执行，这并不意味着线程就会立即运行。

run()方法是线程启动后要进行回调（callback）的方法。

## 5. 编写多线程程序的几种实现方式（换个问法：创建多线程的方式）？

（1）通过继承Thread类

（2）通过实现Runnable接口（推荐使用，因为Java中是单继承，一个类只有一个父类，若继承了Thread类，就无法在继承其它类，显然实现Runnable接口更为灵活）

（3）通过实现Callable接口（Java 5之后）

## 6.解决多线程安全问题的几种方式?

（1）同步代码块:

在代码块声明上 加上synchronized

synchronized (锁对象) {

可能会产生线程安全问题的代码

}

**同步代码块中的锁对象可以是任意的对象；但多个线程时，要使用同一个锁对象才能够保证线程安全。**

（2）同步方法：

在方法声明上加上synchronized

public synchronized void method(){

可能会产生线程安全问题的代码

}

**同步方法中的锁对象是 this**

静态同步方法: 在方法声明上加上static synchronized

**静态同步方法中的锁对象是 类名.class**

（3）同步锁

Lock接口提供了与synchronized关键字类似的同步功能，但需要在使用时手动获取锁和释放锁。

## 7.sleep()和wait()有什么区别？

**1、**每个对象都有一个锁来控制同步访问，Synchronized关键字可以和对象的锁交互，来实现同步方法或同步块。

**sleep()方法**正在执行的线程主动让出CPU（然后CPU就可以去执行其他任务），在sleep指定时间后CPU再回到该线程继续往下执行(注意：sleep方法只让出了CPU，而并不会释放同步资源锁！！！)；

**wait()方法**则是指当前线程让自己暂时退让出同步资源锁，以便其他正在等待该资源的线程得到该资源进而运行，只有调用了notify()方法，之前调用wait()的线程才会解除wait状态，可以去参与竞争同步资源锁，进而得到执行。（注意：notify的作用相当于叫醒睡着的人，而并不会给他分配任务，就是说notify只是让之前调用wait的线程有权利重新参与线程的调度）；

**2、sleep()方法**可以在任何地方使用；**wait()方法**则只能在同步方法或同步块中使用；

**3、sleep()**是线程线程类（Thread）的方法，调用会暂停此线程指定的时间，但监控依然保持，不会释放对象锁，到时间自动恢复；**wait()**是Object的方法，调用会放弃对象锁，进入等待队列，待调用notify()/notifyAll()唤醒指定的线程或者所有线程，才会进入锁池，再次获得对象锁才会进入运行状态；

## 8.线程的五种状态和转换方式？

线程从创建、运行到结束总是处于下面五个状态之一：新建状态、就绪状态、运行状态、阻塞状态及死亡状态。

### 新建状态

当用new操作符创建一个线程时。此时程序还没有开始运行线程中的代码。

### 就绪状态

一个新创建的线程并不自动开始运行，要执行线程，必须调用线程的start()方法。当线程对象调用start()方法即启动了线程，start()方法创建线程运行的系统资源，并调度线程运行run()方法。当start()方法返回后，线程就处于就绪状态。

处于就绪状态的线程并不一定立即运行run()方法，线程还必须同其他线程竞争CPU时间，只有获得CPU时间才可以运行线程。因为在单CPU的计算机系统中，不可能同时运行多个线程，一个时刻仅有一个线程处于运行状态。因此此时可能有多个线程处于就绪状态。对多个处于就绪状态的线程是由Java运行时系统的线程调度程序来调度的。

### 运行状态（running）

当线程获得CPU时间后，它才进入运行状态，真正开始执行run()方法。

### 阻塞状态（blocked）

线程运行过程中，可能由于各种原因进入阻塞状态：

①线程通过调用sleep方法进入睡眠状态；

②线程调用一个在I/O上被阻塞的操作，即该操作在输入输出操作完成之前不会返回到它的调用者；

③线程试图得到一个锁，而该锁正被其他线程持有；

④线程在等待某个触发条件；

所谓阻塞状态是正在运行的线程没有运行结束，暂时让出CPU，这时其他处于就绪状态的线程就可以获得CPU时间，进入运行状态。

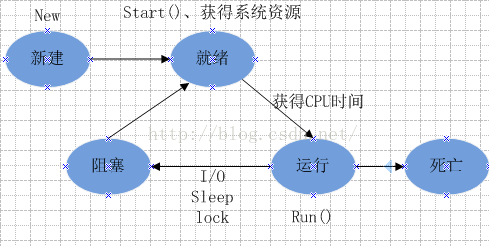
### 死亡状态（dead）

有两个原因会导致线程死亡：

①run方法正常退出而自然死亡；

②一个未捕获的异常终止了run方法而使线程猝死；

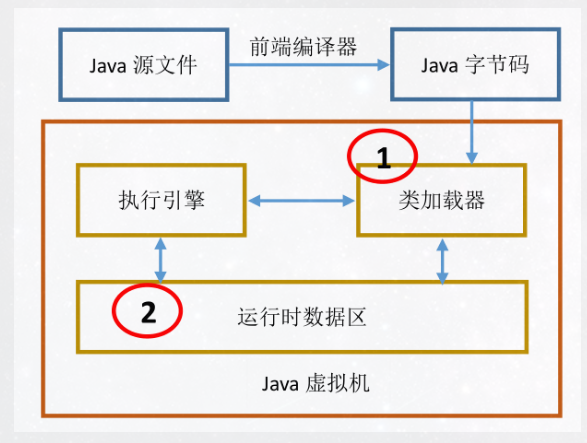
为了确定线程在当前是否存活着（就是要么是可运行的，要么是被阻塞了），需要使用isAlive方法，如果是可运行或被阻塞，这个方法返回true；如果线程仍旧是new状态且不是可运行的，或者线程死亡了，则返回false。



## 9. 什么是死锁？

两个进程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是两个进程 都陷入了无限的等待中。

## 10. java虚拟机的运行机制？



详细请见：

http://www.cnblogs.com/zhanglei93/p/6590609.html

## 11. 简述下你了解的设计模式？

### 一、设计模式的分类（加粗为常用）

设计模式的六大原则

* 开闭原则（Open Closed Principle，OCP）
* 里氏代换原则（Liskov Substitution Principle，LSP）
* 依赖倒转原则（Dependency Inversion Principle，DIP）
* 接口隔离原则（Interface Segregation Principle，ISP）
* 合成/聚合复用原则（Composite/Aggregate Reuse Principle，CARP）
* 最小知识原则（Principle of Least Knowledge，PLK，也叫迪米特法则）

总体来说设计模式分为三大类：

**创建型模式，**共五种**：工厂方法模式**、抽象工厂模式、**单例模式**、建造者模式、原型模式。

**结构型模式，**共七种**：**适配器模式、**装饰器模式**、代理模式、外观模式、桥接模式、组合模式、享元模式。

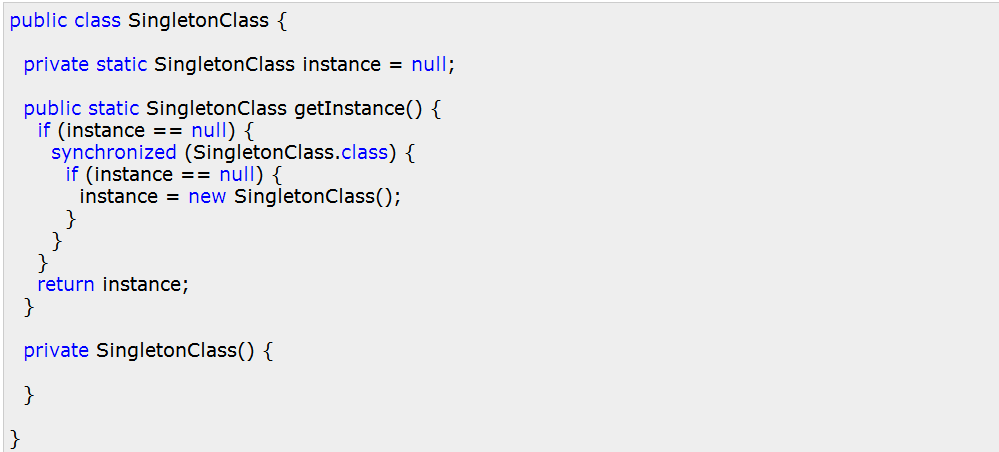
**行为型模式，共十一种：**策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

### 二、单例模式

1、单例类只能有一个实例。

2、单例类必须自己创建自己的唯一实例。

3、单例类必须给所有其他对象提供这一实例。

**线程安全的饿汉式**

### 三、工厂模式

**意图：**定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，工厂模式使其创建过程延迟到子类进行。

**主要解决：**主要解决接口选择的问题。

**何时使用：**我们明确地计划不同条件下创建不同实例时。

**如何解决：**让其子类实现工厂接口，返回的也是一个抽象的产品。

**关键代码：**创建过程在其子类执行。

**工厂模式具体详见链接：**

http://blog.csdn.net/jason0539/article/details/23020989

### 四、装饰者模式

**意图：**装饰者模式通过组合的方式扩展对象的特性，这种方式允许我们在任何时候对对象的功能进行扩展甚至是运行时扩展。

**主要解决：**一般的，我们为了扩展一个类经常使用继承方式实现，由于继承为类引入静态特征，并且随着扩展功能的增多，子类会很膨胀。

**何时使用：** 在不想增加很多子类的情况下扩展类。

**如何解决：**将具体功能职责划分，同时继承装饰者模式。。

**关键代码：**1、Component 类充当抽象角色，不应该具体实现。 2、修饰类引用和继承 Component 类，具体扩展类重写父类方法。

**装饰者模式具体详见链接：**

<http://www.runoob.com/design-pattern/decorator-pattern.html>

# 第二部分：Java web

## 1. http协议有哪些组成部分？

**请求部分：**请求行，请求头，请求体



**响应部分：**响应行，响应头，响应体



常用方法（全部大写）：

**GET请求**获取Request-URI所标识的资源，例如：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用GET方法向服务器获取资源  
**POST**在Request-URI所标识的资源后附加新的数据，例如：要求被请求服务器接受附在请求后面的数据，常用于提交表单



**常见状态码，状态描述：**

  
**200** OK      //客户端请求成功

**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

## 2. Get和Post的区别？

**Get请求方式：**地址栏里会显示我们提交的数据（不安全），并且地址栏中支持提交少量数据，请求的数据存在请求行中

**Post请求方式：**地址栏里不显示我们提交的数据信息（相对安全），可以提交大量数据，请求的数据存在请求正文中

## 3. cookie和session的区别？

**共同点：**cookie和session都是用来跟踪浏览器用户身份的会话方式

**区别：**cookie数据保存在客户端，保存数据不安全且存储数据量有限；

session数据保存在服务器端，保存数据安全且存储数据量大，session是基于cookie进行信息处理的

二者的详细流程请访问：<http://blog.csdn.net/axin66ok/article/details/6175522>

## 4.什么是ajax，为什么要用ajax（浅谈你对ajax的认识），ajax的最大特点？

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：**

1. 通过异步模式，提升了用户体验  
2. 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  
3. 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：http://www.jb51.net/article/93258.htm

## 5.Cookie和Session以及Servlet的生命周期

**Cookie**：

cookie的生命周期是累计的，从创建时，就开始计时，20分钟后，cookie生命周期结束。

**Session:**

session的生命周期是间隔的，从创建时，开始计时如在20分钟，没有访问session，那么session生命周期被销毁

但是，如果在20分钟内（如在第19分钟时）访问过session，那么，将重新计算session的生命周期

注意：关机会造成session生命周期的结束，但是对cookie没有影响

**Servlet:**

init（）：在Servlet的生命周期中，仅执行一次init()方法。它是在服务器装入Servlet时执行的，负责初始化Servlet对象。可以配置服务器，以在启动服务器或客户机首次访问Servlet时装入Servlet。无论有多少客户机访问Servlet，都不会重复执行init（）。

service（）：它是Servlet的核心，负责响应客户的请求。每当一个客户请求一个HttpServlet对象，该对象的Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对象和一个“响应”（ServletResponse）对象作为参数。在HttpServlet中已存在Service()方法。默认的服务功能是调用与HTTP请求的方法相应的do功能。

destroy（）： 仅执行一次，在服务器端停止且卸载Servlet时执行该方法。当Servlet对象退出生命周期时，负责释放占用的资源。一个Servlet在运行service()方法时可能会产生其他的线程，因此需要确认在调用destroy()方法时，这些线程已经终止或完成。

详细代码请看（封装）：<http://blog.csdn.net/qq_29542611/article/details/52426006>

## 6. 说下你熟悉的常用linux命令？

列出文件列表：**ls 【参数 -a -l】**

创建目录和移除目录：**mkdir rmdir**

用于显示文件后几行内容：**tail**

打包：**tar -xvf**

打包并压缩：**tar -zcvf**

查找字符串：**grep**

显示当前所在目录：**pwd**

创建空文件：**touch**

编辑器：**vim vi**

# 第三部份：数据库

## 一：Mysql数据库

### 1、DELETE与TRUNCATE区别

DELETE语句执行删除的过程是每次从表中删除一行，并且同时将该行的删除操作作为事务记录在日志中保存以便进行进行回滚操作，不清空AUTO\_INCREMENT记录数

Truncate则直接将表删除并重新建表，不会把单独的删除操作记录记入日志保存，删除行是不能恢复的,AUTO\_INCREMENT将置为0，效率比DELETE高

### 2、存储过程

#### 2.1 概念

存储过程是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行它。

#### 2.2创建存储过程怎么写？

“pr\_add” 是个简单的 MySQL 存储过程，这个MySQL 存储过程有两个 int 类型的输入参数 “a”、“b”，返回这两个参数的和。

1）**drop procedure if exists pr\_add;**

2）计算两个数之和

**create procedure pr\_add   (   a int,   b int   )   begin   declare c int;**

**if a is null then   set a = 0;**

**end if;**

**if b is null then   set b = 0;**

**end if;**

**set c = a + b;**

**select c as sum;**

### 3、索引

#### 3.1概念

索引是对数据库中一或多个列值的排序，帮助数据库高效获取数据的数据结构。假如我们用类比的方法，数据库中的索引就相当于书籍中的目录一样，当我们想找到书中的摸个知识点，我们可以直接去目录中找而不是在书中每页的找，但是这也抛出了索引的一个缺点，在对数据库修改的时候要修改索引到导致时间变多。

#### 3.2分类

普通索引 唯一索引 主键索引 全文索引

#### 3.3优点

* + 加快检索速度
  + 唯一索引确保每行数据的唯一性
  + 在使用索引的过程可以优化隐藏器，提高系统性能

#### 3.4缺点

* + 插入删除 修改 维护速度下降
  + 占用物理和数据空间

### 4、事务

#### 4.1 应用场景：存在并发数据访问时才需要事务

#### 4.2 ACID四大特性

* 1. **原子性(Atomicity):** 整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。任何一项操作的失败都会导致整个事务的失败。
  2. **一致性(Correspondence):** 在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
  3. **隔离性(Isolation):** 并发执行的事务彼此无法看到对方的中间状态。
  4. **持久性(Durability):** 在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

#### 4.3事务中的问题与隔离级别

* **问题**

**脏读**: 一个事务读取到另一个事务未提交的数据;

**不可重复读**:一个事务中两次查询的数据不一致 --> 一个事务读到了另一个事务 已经提交数据(update操作)

**虚读(幻读)**:一个事务中两次查询的数据不一致 --> 一个事务读到了另一个事务 已经提交数据(insert操作)

* **隔离级别(安全从低到高,性能从高到低)**
  1. **读未提交：也叫脏读，是事务可以读取其它事务未提交的数据。**

--> 未解决任何问题;

* 1. **读已提交：在事务未提交之前所做的修改其它事务是不可见的**

--> 解决脏读问题;

* 1. **可重复读：保证同一个事务中的多次相同的查询的结果是一致的**

--> 解决脏读,不可重复读问题;

* 1. **可串行化： 保证读取的范围内没有新的数据插入，比如事务第一次查询得到某个范围的数据，第二次查询也同样得到了相同范围的数据，中间没有新的数据插入到该范围中**

--> 解决脏读,不可重复读,虚读(幻读)问题;

常用数据库默认隔离级别：

MySQL: 可重复读

Oracle: 读已提交

SQLServer: 读已提交

#### 4.4如何进行事务管理

Connection 提供了事务处理的方法，通过调用 setAutoCommit(false)可以设置手动提交事务；当事务完成后用 commit()显式提交事务；如果在事务处理过程中发生异常则通过 rollback()进行事务回滚。除此之外，从 JDBC 3.0 中还引入了 Savepoint（保存点）的概念，允许通过代码设置保存点并让事务回滚到指定的保存点

### 5、**mysql，oracle，sql server三者的区别**

#### 5.1 mysql

**优点**：  
体积小、速度快、总体拥有成本低，开源；支持多种操作系统；是开源数据库，提供的接口支持多种语言连接操作。

**缺点**：  
不支持热备份；  
MySQL最大的缺点是其安全系统，主要是复杂而非标准，另外只有到调用mysqladmin来重读用户权限时才发生改变；  
没有一种存储过程(Stored Procedure)语言，这是对习惯于企业级数据库的程序员的最大限制；  
MySQL的价格随平台和安装方式变化。Linux的MySQL如果由用户自己或系统管理员而不是第三方安装则是免费的，第三方案则必须付许可费。Unix或Linux 自行安装 免费 、Unix或Linux 第三方安装 收费；

#### 5.2 oracle

**优点**：  
 **开放性：**Oracle 能在所有主流平台上运行（包括 windows）完全支持所有工业标准， 采用完全开放策略，使客户选择适合解决方案；

**可伸缩性,并行性：**Oracle 并行服务器通过使组结点共享同簇工作来扩展windownt能 力，提供高用性和高伸缩性簇解决方案。

**安全性**：获得最高认证级别的ISO标准认证。

**性能**：Oracle 性能高 保持开放平台下TPC-D和TPC-C世界记录；

**客户端支持及应用模式**：Oracle 多层次网络计算支持多种工业标准用ODBC、JDBC、OCI 等网络客户连接   
 **使用风险**：Oracle 长时间开发经验完全向下兼容得广泛应用地风险低 。

**缺点**：  
 对硬件的要求很高；  
 价格比较昂贵；  
 管理维护麻烦一些；  
 操作比较复杂，需要技术含量较高；

#### 5.3 sqlserver

**优点**：

易用性、适合分布式组织的可伸缩性、用于决策支持的数据仓库功能、与许多其他服务 器软件紧密关联的集成性、良好的性价比等；

SQLServer是一个具备完全Web支持的数据库产品，提供了对可扩展标记语言 (XML) 的核心支持以及在Internet上和防火墙外进行查询的能力；

**缺点：**SQL Server 只能windows上运行，没有丝毫开放性操作系统。

**伸缩性并行性** ：数据卷伸缩性有限；

**安全性**：没有获得任何安全证书。

**性能** ：SQL Server 多用户时性能佳 ；

**客户端支持及应用模式**： 客户端支持及应用模式。只支持C/S模式，SQL Server C/S 结构只支持windows客户用ADO、DAO、OLEDB、ODBC连接；

**使用风险**：SQL server 完全重写代码经历了长期测试，需要时间来证明并十分兼容；

## 三：Redis数据库（redis的持久化机制，redis的集群搭建详见电商项目知识点4）

### 1、什么是redis?

[Redis](http://lib.csdn.net/base/redis" \t "http://www.cnblogs.com/jiahaoJAVA/p/_blank" \o "Redis知识库)是一个基于内存的高性能key-value[数据库](http://lib.csdn.net/base/mysql" \t "http://www.cnblogs.com/jiahaoJAVA/p/_blank" \o "MySQL知识库)。

应用场景：

**（**1）会话缓存（Session Cache）

（2）全页缓存（FPC）

（3）队列

（4）排行榜/计数器

（5）发布/订阅

### 2、使用redis有哪些好处

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

### 3、Reids的特点？

Redis本质上是一个Key-Value类型的内存数据库，很像memcached，整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。  
Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种[数据结构](http://lib.csdn.net/base/datastructure" \t "http://www.cnblogs.com/jiahaoJAVA/p/_blank" \o "算法与数据结构知识库),此外单个value的最大限制是1GB，不像 memcached只能保存1MB的数据，另外Redis也可以对存入的Key-Value设置expire时间。

Redis的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

### 4、为什么redis需要把所有数据放到内存中？

Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度会严重影响redis的性能。如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

### 5、Redis常见的性能问题解决？

(1) Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件

(2) 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变

### 6、redis相比memcached有哪些优势？

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

## 四：数据库语句优化

**1. 对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。**

**2. 应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描**

**3.** **应尽量避免在 where 子句中使用!=或<>操作符，否则引擎将放弃使用索引而进行全表扫描。**

**4.** **应尽量避免在 where 子句中使用or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描**

**5. in 和 not in 也要慎用，否则会导致全表扫描**

**6. 索引并不是越多越好，索引固然可 以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为insert或update时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有必要**

**7. 查询结果不要用\*来查询所有字段，要明确指明结果字段**

**8. 根据查询条件，建立索引，如果查询条件不止一个时，使用组合索引**

**9. 在查询条件表达式的左侧尽量不要使用函数，否则索引失效**

**10. 如果有like话，尽量避免%xxx%两侧都有%的条件，单侧%可以使用索引，多侧不可以**

**11.建立索引时字段不能有null值**

## 五：数据库优化

### 1、Mysql数据库优化有哪些？

**(1) EXPLAIN 你的 SELECT 查询**

**(2)** **当只要一行数据时使用 LIMIT 1**

**(3)** **使用 ENUM 而不是 VARCHAR**

**(4)** **固定长度的表会更快**

**(5)** **分库分表**

### 2、Oracle 数据库优化有哪些？

**调整数据结构的设计。**

**调整应用程序结构设计。**

**调整数据库SQL语句。**

**调整服务器内存分配。**

**调整硬盘I/O。**

**调整操作系统参数。**

# 第四部份：框架

## 二：Spring

### 1. 谈谈你理解的spring的工作流程的理解？

* 创建配置文件applicationContext.xml
* 编写配置文件（加入一些对象的配置信息）
* Spring内部采用工厂模式，配合xml解析+反射技术，可以根据用户的配置，生成相应的对象
* 工厂提供一个getBean方法，从工厂中获取对象
* 操作对象的方法，属性

### 2. 说说spring对象创建的三种方式。

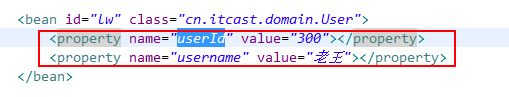
1.无参构造2.实例工厂 3.静态工厂<**关于创建方式的描述，同学们自己组织一下自己的理解进行添加**>

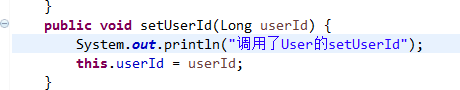
### 3 不使用注解的情况下，如果给对象注入值的话，你知道的有几种方式？常用的是哪些？

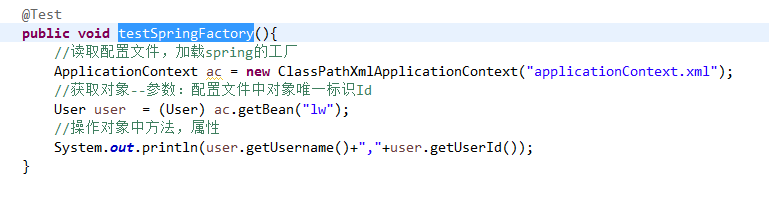
1. set注值(**重点**) 2.构造器注值 3.P名称空间注值 4.Spel注值 5.注入java复杂类型6.注入对象类型(**重点**)aa

**我常用的是set注值和对象类型注值。**

**Set** 我们通过配置文件给对象(User)赋值,就相当于我们使用对象里边属性的set方法给对象设置值。







**对象类型注值** 使用ref表示注入一个已经存在的对象（默认还是调用set方法注入）

### 4. Spring有注解方式和xml配置两种，两者有什么区别？你常用spring注解有哪些？

我在公司主要使用的是注解开发，能够提高开发效率，在配置文件中把相关的注解扫描配置好，我们只要在自己写的类或者代码上边添加对应的注解就把事情解决了。

注解开发

1）创建对象

导入jar包（IOC基本）spring-aop-4.2.4.jar

导入约束context

开启注解扫描<context componect-scan base-package=”包名”>

在要创建的对象上使用注解@Component @Repository @Service @Controller

如果要产生对象是多例@Scope(value=”prototype”)

2)注入对象

在service有dao的属性

在dao的属性上加注解 @Autowired 按类型注入

@Reosource（name=””） 按bean的id注入

### 5. 谈谈spring的事务管理

spring提供的事务管理可以分为两类：编程式的和声明式的。编程式的，比较灵活，但是代码量大，存在重复的代码比较多；声明式的比编程式的更灵活。

编程式 主要使用transactionTemplate。省略了部分的提交，回滚，一系列的事务对象定义，需注入事务管理对象.

声明式：使用TransactionProxyFactoryBean:PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED,readOnly

围绕Poxy的动态代理 能够自动的提交和回滚事务

### 6. spring原理

①IoC(Inversion of control): 控制反转，依赖注入

1） IoC：

概念：控制权由对象本身转向容器；由容器根据配置文件去创建实例并创建各个实例之间的依赖关系

2） 依赖IoC容器负责管理bean，有两种，一种是BeanFactory，另一种是ApplicationContext，但是ApplicationContext继承与BeanFactory。

核心：bean工厂；在Spring中，bean工厂创建的各个实例称作bean

②AOP(Aspect-Oriented Programming): 面向方面编程

1） 代理的两种方式：

静态代理：

? 针对每个具体类分别编写代理类；

? 针对一个接口编写一个代理类；

动态代理：

针对一个方面编写一个InvocationHandler，然后借用JDK反射包中的Proxy类为各种接口动态生成相应的代理类

2） AOP的主要原理：动态代理实现：有两种：JDK Proxy和Cglib，Spring规定对于有接口的类用JDK Proxy，对于无接口和抽象类用Cglib，虽然Cglib均可以代理，但是Cglib复杂，效率低。但是Cglib有例外，就是代理的类中不能是final修饰的类或者类中有final方法。

### 7.使用Spring有什么好处？

◆Spring能有效地组织你的中间层对象,无论你是否选择使用了EJB。如果你仅仅使用了 Struts或其他的包含了J2EE特有APIs的framework，你会发现Spring关注了遗留下的问题。

◆Spring能消除在许多工程上对Singleton的过多使用。根据我的经验，这是一个主要的问题，它减少了系统的可测试性和面向对象特性。

◆Spring能消除使用各种各样格式的属性定制文件的需要,在整个应用和工程中，可通过一种一致的方法来进行配置。曾经感到迷惑，一个特定类要查找迷幻般的属性关键字或系统,为此不得不读Javadoc乃至源编码吗？有了Spring，你可很简单地看到类的avaBean属性。倒置控制的使用(在下面讨论)帮助完成这种简化。

◆Spring能通过接口而不是类促进好的编程习惯，减少编程代价到几乎为零。

◆Spring被设计为让使用它创建的应用尽可能少的依赖于他的APIs。在Spring应用中的大多数业务对象没有依赖于Spring。

◆使用Spring构建的应单元测试。

◆Spring能使EJB的使用成为一个实现选择,而不是应用架构的必然选择。你能选择用POJOs或local EJBs来实现业务接口，却不会影响调用代码。

◆Spring帮助你解决许多问题而无需使用EJB。Spring能提供一种EJB的替换物，它们适于许多web应用。例如Spring能使用AOP提供声明性事务而不通过使用EJB容器，如果你仅仅需要与单个的数据库打交道，甚至不需要JTA实现。

■Spring为数据存取提供了一致的框架,不论是使用JDBC或O/R mapping产品（如Hibernate）。

Spring确实使你能通过最简单可行的解决办法解决你的问题。这些特性是有很大价值的。

总结起来，Spring有如下优点：

◆低侵入式设计，代码污染极低

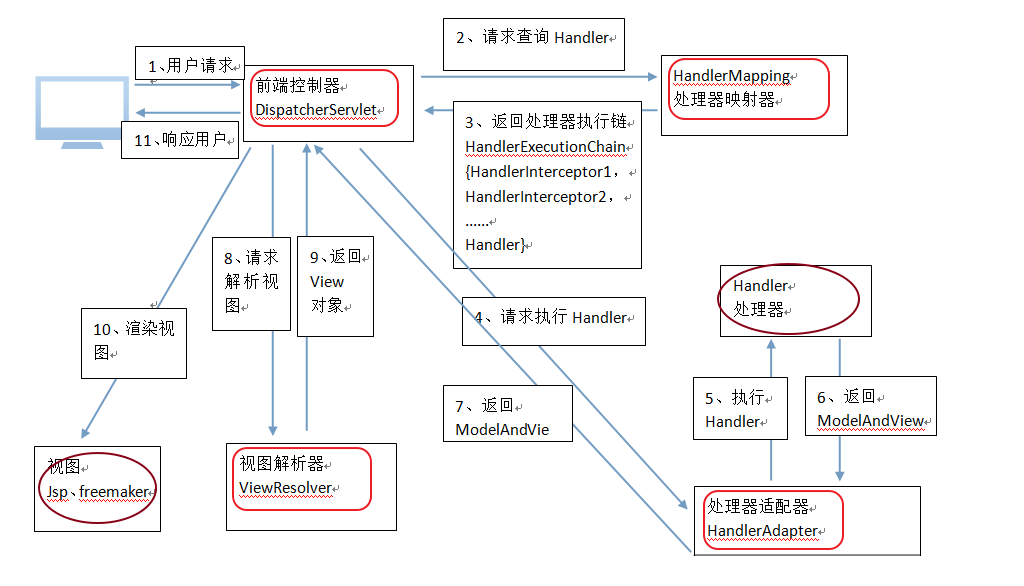
◆独立于各种应用服务器，可以真正实现Write Once,Run Anywhere的承诺

◆Spring的DI机制降低了业务对象替换的复杂性

◆Spring并不完全依赖于Spring，开发者可自由选用Spring框架的部分或全部

## 四：SpringMVC

### 1、SpringMVC的执行流程是什么？请简单阐述一下。



### 2、你常用的SpringMVC注解有哪些？如何开启SpringMVC的注解扫描？

常用的注解有：@requestMapping 用于请求url映射。

@RequestBody注解实现接收http请求的json数据，将json数据转换为java对象。

@ResponseBody注解实现将controller方法返回对象转化为json响应给客户。

**其他的注解希望学生根据自己理解情况自行补充。**

### 3、如何开启注解处理器和适配器的配置？

我们在项目中一般会在springmvc.xml中通过开启 <mvc:annotation-driven>来实现注解处理器和适配器的开启。

### 4、使用springMVC框架的时候，如何解决post和get的乱码问题？

**解决post请求乱码：**我们可以在web.xml里边配置一个CharacterEncodingFilter过滤器。设置为utf-8.

**解决get请求的乱码：**有两种方法。对于get请求中文参数出现乱码解决方法有两个：

1.修改tomcat配置文件添加编码与工程编码一致。

2.另外一种方法对参数进行重新编码 String userName = New String(Request.getParameter(“userName”).getBytes(“ISO8859-1”), “utf-8”);

### 5、SpringMVC与struts2的区别？

**1、**springmvc的入口是一个servlet即前端控制器，而struts2入口是一个filter过虑器。

**2、**springmvc是基于方法开发，传递参数是通过方法形参，可以设计为单例或多例(建议单例)，struts2是基于类开发，传递参数是通过类的属性，只能设计为多例。

**3、**Struts采用值栈存储请求和响应的数据，通过OGNL存取数据， springmvc通过参数解析器是将request对象内容进行解析成方法形参，将响应数据和页面封装成ModelAndView对象，最后又将模型数据通过request对象传输到页面。 Jsp视图解析器默认使用jstl。

**4、**Struts2有漏洞，springmvc目前还没有漏洞出现。如果使用struts2，建议下载最新包。

## 五：Mybatis

### 1、MyBatis编程步骤是什么样的？

1、创建SqlSessionFactory

2、通过SqlSessionFactory创建SqlSession

3、通过sqlsession执行数据库操作

4、调用session.commit()提交事务

5、调用session.close()关闭会话

提示:MyBatis与Hibernate开发流程基本一致

### 2、Mybatis和Hibernate的区别。

**mybatis：**

1. 入门简单，即学即用，提供了数据库查询的自动对象绑定功能，而且延续了很好的SQL使用经验，对于没有那么高的对象模型要求的项目来说，相当完美。

2. 可以进行更为细致的SQL优化，可以减少查询字段。

3. 缺点就是框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定[代码](http://www.xuebuyuan.com/" \t "http://www.xuebuyuan.com/_blank" \o "代码)，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改。

4. 二级缓存机制不佳。

**hibernate：**

1. 功能强大，数据库无关性好，O/R映射能力强，如果你对Hibernate相当精通，而且对Hibernate进行了适当的封装，那么你的项目整个持久层代码会相当简单，需要写的代码很少，开发速度很快，非常爽。

2. 有更好的二级缓存机制，可以使用第三方缓存。

3. 缺点就是学习门槛不低，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡取得平衡，以及怎样用好Hibernate方面需要你的经验和能力都很强才行。

**为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？**

Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。

### 5、使用MyBatis的mapper接口调用时有哪些要求？

①  Mapper接口方法名和mapper.xml中定义的每个sql的id相同

②  Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同

③  Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同

④  Mapper.xml文件中的namespace即是mapper接口的类路径。

注意:Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

不同的Xml映射文件，如果配置了namespace，那么id可以重复；如果没有配置namespace，那么id不能重复；毕竟namespace不是必须的，只是最佳实践而已。原因就是namespace+id是作为Map<String, MappedStatement>的key使用的，如果没有namespace，就剩下id，那么，id重复会导致数据互相覆盖。

### 6、mysql中mybatis如何获取自增型主键值？

<insert id="insert" parameterType="cn.\*\*\*.beans.LogObject" >

<selectKey resultType="java.lang.Integer" order="BEFORE" keyProperty="id">

SELECT LAST\_INSERT\_ID()

</selectKey>

insert into user(userName,password,comment)

values(#{userName},#{password},#{comment})

</insert>

注意：selectKey中order属性的值是不同的，BEFORE先选择主键，设置keyProperty的值然后执行插入语句。

AFTER是先执行插入语句，然后执行selectKey。

<insert id="insertAndGetId" useGeneratedKeys="true" keyProperty="id" parameterType="com.chenzhou.mybatis.User">

insert into user(userName,password,comment)

values(#{userName},#{password},#{comment})

</insert>

### 7、在mapper中如何传递多个参数?

//DAO层的函数

Public UserselectUser(String name,String area);

//对应的xml,#{0}代表接收的是dao层中的第一个参数，#{1}代表dao层中第二参数，更多参数一致往后加即可。

<select id="selectUser"resultMap="BaseResultMap">

select \* fromuser\_user\_t whereuser\_name = #{0} anduser\_area=#{1}

</select>

使用 @param 注解:

public interface usermapper {

user selectuser(@param(“username”) string username,

@param(“hashedpassword”) string hashedpassword);

}

然后,就可以在xml像下面这样使用(推荐封装为一个map,作为单个参数传递给mapper):

<select id=”selectuser” resulttype=”user”>

select id, username, hashedpassword

from some\_table

where username = #{username}

and hashedpassword = #{hashedpassword}

</select>

### 8、#{}和${}的区别是什么？

#{}是预编译处理，${}是字符串替换。

Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

Mybatis在处理${}时，就是把${}替换成变量的值。

使用#{}可以有效的防止SQL注入，提高系统安全性。

### 9、当实体类中的属性名和表中的字段名不一样 ，怎么办 ？

第1种： 通过在查询的sql语句中定义字段名的别名，让字段名的别名和实体类的属性名一致

第2种： 通过<resultMap>来映射字段名和实体类属性名的一一对应的关系

# 第五部份：电商项目

## 1、Dubbo

### 1. Dubbo是什么？

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架  
其核心部分包含:  
1. 远程通讯: 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式。  
2. 集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。  
3. 自动发现: 基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

### 2. Dubbo能做什么？

1.透明化的远程方法调用，就像调用本地方法一样调用远程方法，只需简单配置，没有任何API侵入。        
2.软负载均衡及容错机制，可在内网替代F5等硬件负载均衡器，降低成本，减少单点。  
3. 服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。  
  
Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。

### 3. Dubbo架构

1. dubbo架构图如下所示：



2 .节点角色说明：

* + 1. Provider: 暴露服务的服务提供方。
    2. Consumer: 调用远程服务的服务消费方。
    3. Registry: 服务注册与发现的注册中心。
    4. Monitor: 统计服务的调用次调和调用时间的监控中心。

1. [Container](http://lib.csdn.net/base/4" \t "_blank" \o "Docker知识库): 服务运行容器。

#### 3.1调用关系说明

1. **服务容器负责启动，加载，运行服务提供者。**
2. **服务提供者在启动时，向注册中心注册自己提供的服务。**
3. **服务消费者在启动时，向注册中心订阅自己所需的服务。**
4. **注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。**
5. **服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。**
6. **服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。**

#### 3.2 dubbo使用方法。

Dubbo采用全[spring](http://lib.csdn.net/base/17" \t "_blank" \o "Java EE知识库)配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可，Dubbo基于Spring的Schema扩展进行加载。如果不想使用Spring配置，而希望通过API的方式进行调用（不推荐），

**服务提供方：**



**服务消费者：**





令牌验证   
防止消费者绕过注册中心访问提供者，在注册中心控制权限，以决定要不要下发令牌给消费者，注册中心可灵活改变授权方式，而不需修改或升级提供者

## 2、FastDFS

### 1. 什么是FastDFS？

FastDFS是一款开源的、分布式文件系统（Distributed File System），由淘宝开发平台部资深架构师余庆开发。该开源项目的主页是 http://code.google.com/p/fastdfs 。可以通过fastdfs.sourceforge.NET 下载。FastDFS孵化平台(ChinaUnix)版块http://bbs.china

unix.net/forum-240-1.html。

FastDFS是一个开源的轻量级分布式文件系统，它对文件进行管理，功能包括：文件存储、文件同步、文件访问（文件上传、文件下载）等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服务，如相册网站、视频网站等等。

FastDFS为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，并注重高可用、高性能等指标，使用FastDFS很容易搭建一套高性能的文件服务器集群提供文件上传、下载等服务。

FastDFS服务端有两个角色：跟踪器（tracker）和存储节点（storage）。跟踪器主要做调度工作，在访问上起负载均衡的作用。

跟踪器和存储节点都可以由一台或多台服务器构成。跟踪器和存储节点中的服务器均可以随时增加或下线而不会影响线上服务。其中跟踪器中的所有服务器都是对等的，可以根据服务器的压力情况随时增加或减少。

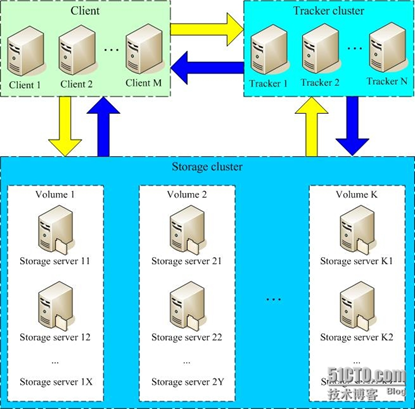
为了支持大容量，存储节点（服务器）采用了分卷（或分组）的组织方式。存储系统由一个或多个卷组成，卷与卷之间的文件是相互独立的，所有卷的文件容量累加就是整个存储系统中的文件容量。一个卷可以由一台或多台存储服务器组成，一个卷下的存储服务器中的文件都是相同的，卷中的多台存储服务器起到了冗余备份和负载均衡的作用。

在卷中增加服务器时，同步已有的文件由系统自动完成，同步完成后，系统自动将新增服务器切换到线上提供服务。

当存储空间不足或即将耗尽时，可以动态添加卷。只需要增加一台或多台服务器，并将它们配置为一个新的卷，这样就扩大了存储系统的容量。

FastDFS中的文件标识分为两个部分：卷名和文件名，二者缺一不可。

。



FastDFS架构图

1)      Tracker cluster中各个tracker server相互独立，不进行相互通信。

2)      Storage cluster中各个storage组（Volume1,Volume2...）相互独立，不进行相互通信，也就是说各个组之间保存的数据是不相同的。但是各个组中的storage server之间是属于互相备份的关系，也就是说storage server之间保存相同的数据。

3)      每个storage server会启动一个单独的线程主动向Tracker cluster中每个tracker server报告其状态信息，包括磁盘使用情况，文件同步情况及文件上传下载次数统计等信息。

### 2.上传时序图



1)      Client通过Tracker server将文件上传到Storage server。

2)      Tracker server向Client返回一台可用的Storage server的IP地址和端口号。

3)      Client直接通过Tracker server返回的IP地址和端口与其中一台Storage server建立连接并进行文件上传。

4)      上传完成，Storage server返回Client一个文件ID，文件上传结束。

### 3.下载时序图

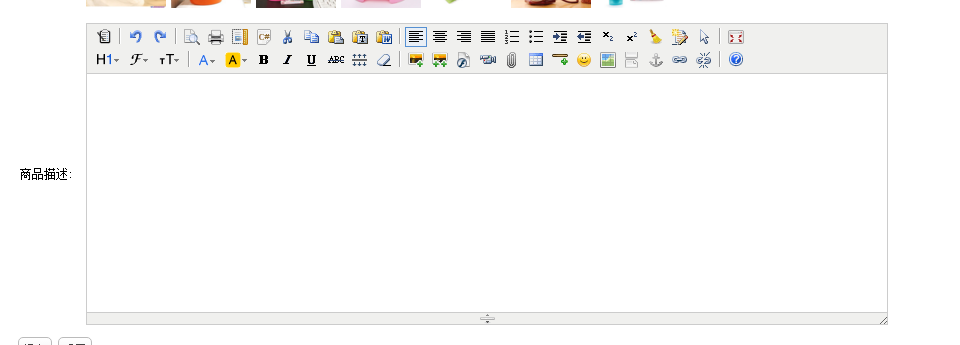


1)      Client通过Tracker server下载指定Storage组中某个Storage server上的某个文件（文件名包括Storage组名称）。

2)      Tracker server向Client返回一台可用的Storage server的IP地址和端口号。

3)      Client直接通过Tracker server返回的IP地址和端口与其中一台Storage server建立连接并进行文件下载。

## 3、富文本编辑器



### 1. KindEditor：

KindEditor使用JavaScript编写，可以无缝的于Java、.NET、PHP、ASP等程序接合。 KindEditor非常适合在CMS、商城、论坛、博客、Wiki、电子邮件等互联网应用上使用，2006年7月首次发布2.0以来，KindEditor依靠出色的用户体验和领先的技术不断扩大编辑器市场占有率，目前在国内已经成为最受欢迎的编辑器之一。

**主要特点：**

1. 体积小，加载速度快，但功能十分丰富。2. 内置自定义range，完美地支持span标记。

3. 基于插件的方式设计，所有功能都是插件，增加自定义和扩展功能非常简单。

4. 修改编辑器风格很容易，只需修改一个CSS文件。

5. 支持大部分主流浏览器，比如IE、Firefox、Safari、Chrome、Opera。

### 2. UEditor：百度编辑器

UEditor是由[百度](http://baike.baidu.com/item/%E7%99%BE%E5%BA%A6" \t "http://baike.baidu.com/_blank)WEB前端研发部开发的所见即所得的开源富文本编辑器，具有轻量、可定制、用户体验优秀等特点。开源基于BSD协议，所有源代码在协议允许范围内可自由修改和使用。[百度](http://baike.baidu.com/item/%E7%99%BE%E5%BA%A6" \t "http://baike.baidu.com/_blank)UEditor的推出，可以帮助不少网站开发者在开发富文本编辑器所遇到的难题，节约开发者因开发富文本编辑器所需要的大量时间，有效降低了企业的开发成本。

**优点：**

1、体积小巧，性能优良，使用简单

2、分层架构，方便定制与扩展

3、满足不同层次用户需求，更加适合团队开发

4、丰富完善的中文文档

5、多个[浏览器](http://baike.baidu.com/item/%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)支持：Mozilla, MSIE, [FireFox](http://baike.baidu.com/item/FireFox" \t "http://baike.baidu.com/_blank), [Maxthon](http://baike.baidu.com/item/Maxthon" \t "http://baike.baidu.com/_blank),Safari 和Chrome

6、更好的使用体验

7、拥有专业QA团队持续支持，已应用在[百度](http://baike.baidu.com/item/%E7%99%BE%E5%BA%A6" \t "http://baike.baidu.com/_blank)各大产品线上

注：纯js开发，跟后台语言没有关系。

## 4、Redis

redis是一个key-value[存储系统](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/Redis/_blank)。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list([链表](https://baike.baidu.com/item/%E9%93%BE%E8%A1%A8" \t "https://baike.baidu.com/item/Redis/_blank))、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些[数据类型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B" \t "https://baike.baidu.com/item/Redis/_blank)都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了[memcached](https://baike.baidu.com/item/memcached" \t "https://baike.baidu.com/item/Redis/_blank)这类key/value存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

### 1. Redis的优点？

* 性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。
* 丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。

原子 – Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。

丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

### 2. Redis的缺点？

是数据库容量受到物理内存的限制,不能用作海量数据的高性能读写,因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

### 3. Redis 持久化

#### 3.1、RDB 持久化：

该机制是指在制定的时间间隔内将内存中的数据集快照写入磁盘。

优点：1.只有一份 rdb 文件，可随时备份

2.比 AOF 文件小，加载效率高

3.只提供 fork 子进程，不阻塞主进程，IO 操作比较少

#### 3.2、AOF 持久化：

该机制将以日志的形式记录服务器所处理的每一个写操作，在 Redis 服务器启动之初 会读取该文件来重新构建数据库，以保证启动后数据库中的数据是完整的。

优点：1.每次改动同步数据安全性好

2.APPEND 方式追加日志，不会对旧日志文件产生影响

#### 3.3、无持久化 ：

我们可以通过配置的方式禁用 Redis 服务器的持久化功能，这样我们就可以将 Redis 视为一个功能加强版的 memcached 了

#### 3.4、同时应用 AOF 和 RDB

### 4、Redis 集群

群指的是将几台服务器集中在一起，实现同一业务

1.目的：高可用、负载均衡、易扩展、数据安全、性能提升

2. 技术：集群地址（虚拟 IP）、网络通信（监控消息）

3. 功能：负载均衡、读写分离、故障转移

### 5、Memcached是什么？

Memcached 是一个高性能的分布式内存对象缓存系统，用于动态Web应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高动态、数据库驱动网站的速度。Memcached基于一个存储键/值对的[hashmap](http://baike.baidu.com/item/hashmap" \t "http://baike.baidu.com/item/_blank)。其[守护进程](http://baike.baidu.com/item/%E5%AE%88%E6%8A%A4%E8%BF%9B%E7%A8%8B" \t "http://baike.baidu.com/item/_blank)（daemon ）是用[C](http://baike.baidu.com/item/C/7252092" \t "http://baike.baidu.com/item/_blank)写的，但是客户端可以用任何语言来编写，并通过memcached协议与守护进程通信。

**缺点**：1：memcached缺乏认证以及安全管制，这代表应该将memcached服务器放置在防火墙后。

2：memcached的API使用32位元的[循环冗余校验](http://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E5%86%97%E4%BD%99%E6%A0%A1%E9%AA%8C" \t "http://baike.baidu.com/item/_blank)（CRC-32）计算键值后，将资料分散在不同的机器上。当表格满了以后，接下来新增的资料会以LRU机制替换掉。由于memcached通常只是当作快取系统使用，所以使用memcached的应用程式在写回较慢的系统时（像是后端的数据库）需要额外的程式码更新memcached内的资料

## 5、MQ

### 1. RabbitMQ：

RabbitMQ是实现AMQP（高级消息队列协议）的消息中间件的一种，最初起源于金融系统，用于在分布式系 统中存储转发消息，在易用性、扩展性、高可用性等方面表现不俗。消息中间件主要用于组件之间的解耦， 消息的发送者无需知道消息使用者的存在，反之亦然.

RabbitMQ 本身支持很多的协议：AMQP，XMPP, SMTP, STOMP，也正是如此，使的它变的非常重量级，更适 合于企业级的开发。同时实现了一个经纪人(Broker)构架，这意味着消息在发送给客户端时先在中心队列 排队。对路由(Routing)，负载均衡(Load balance)或者数据持久化都有很好的支持。

#### RabbitMQ 的优点（适用范围）

1、基于 erlang 语言开发具有高可用高并发的优点，适合集群服务器。

2、健壮、稳定、易用、跨平台、支持多种语言、文档齐全。   
3、 有消息确认机制和持久化机制，可靠性高。

4、开源 其他 MQ 的优势： 1.2.1 Apache ActiveMQ 曝光率最高，但是可能会丢消息。 5、ZeroMQ 延迟很低、支持灵活拓扑，但是不支持消息持久化和崩溃恢复。

### 2. AcitveMQ

#### 1. 什么是ActiveMQ？

ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现,尽管JMS规范出台已经是很久的事情了,但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

**主要特点：**

1. 多种语言和协议编写客户端。语言: Java, C, C++, C#, Ruby, Perl, Python, PHP。应用协议: OpenWire,Stomp REST,WS Notification,XMPP,AMQP

2. 完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)

3. 对Spring的支持,ActiveMQ可以很容易内嵌到使用Spring的系统里面去,而且也支持Spring2.0的特性

4. 通过了常见J2EE服务器(如 Geronimo,JBoss 4, GlassFish,WebLogic)的测试,其中通过JCA 1.5 resource adaptors的配置,可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上

5. 支持多种传送协议:in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA

6. 支持通过JDBC和journal提供高速的消息持久化

7. 从设计上保证了高性能的集群,客户端-服务器,点对点

8. 支持Ajax

9. 支持与Axis的整合

10. 可以很容易得调用内嵌JMS provider,进行测试

#### 2. ActiveMQ的消息形式

对于消息的传递有两种类型：

**一种是点对点的**，即一个生产者和一个消费者一一对应；

**另一种是发布/订阅模式**，即一个生产者产生消息并进行发送后，可以由多个消费者进行接收。

**点到点模式：**如果消息发送不成功此 消息默认会保存到 activemq 服务端知道有消费者将其消费，所以此时消息是不会丢失的。

**发布订阅模式的通信方式：**默认情况下只通知一次，如果接收不到此消息就没有了。这种场景只适 用于对消息送达率要求不高的情况。如果要求消息必须送达不可以丢失的话，需要配置持久订阅。每个订阅端定义一个 id，在订阅是向 activemq 注册。发布消息和接收消息时需要配置发送模式为持久化。此时 如果客户端接收不到消息，消息会持久化到服务端，直到客户端正常接收后为止。

JMS定义了五种不同的消息正文格式，以及调用的消息类型，允许你发送并接收以一些不同形式的数据，提供现有消息格式的一些级别的兼容性。

　　· StreamMessage -- Java原始值的数据流

　　· MapMessage--一套名称-值对

　　· TextMessage--一个字符串对象

　　· ObjectMessage--一个序列化的 Java对象

　　· BytesMessage--一个字节的数据流

#### 3. AcitveMQ 的作用、原理？（生产者。消费者。 p2p、订阅实现流程）

Activemq 的作用就是系统之间进行通信。当然可以使用其他方式进行系统间通信，如果使用 Activemq 的 话可以对系统之间的调用进行解耦，实现系统间的异步通信。原理就是生产者生产消息，把消息发送给 activemq。Activemq 接收到消息，然后查看有多少个消费者，然后把消息转发给消费者，此过程中生产者 无需参与。消费者接收到消息后做相应的处理和生产者没有任何关系。

#### 4. ActiveMQ 在项目中应用场景？

Activemq 在项目中主要是完成系统之间通信，并且将系统之间的调用进行解耦。例如在添加、修改商品信 息后，需要将商品信息同步到索引库、同步缓存中的数据以及生成静态页面一系列操作。在此场景下就可 以使用 activemq。一旦后台对商品信息进行修改后，就向 activemq 发送一条消息，然后通过 activemq 将 消息发送给消息的消费端，消费端接收到消息可以进行相应的业务处理。

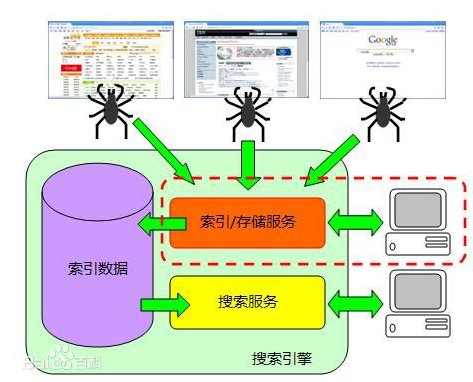
#### 5. ActiveMQ 如果数据提交不成功怎么办？

假如现在我手里有一个很重要的消息的，想要发给一个人，但是很不幸，消息发送失败了。这时候怎么办呢怎么解决这种尴尬的情况，这时候我们可以利用activeMQ的消息重发机制（RedeliveryPolicy）  来解决这个烦人的问题。

## 6、Solr

### 1. Solr是什么

Solr是一个独立的[企业级搜索](https://baike.baidu.com/item/%E4%BC%81%E4%B8%9A%E7%BA%A7%E6%90%9C%E7%B4%A2" \t "https://baike.baidu.com/item/Solr/_blank)应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果。

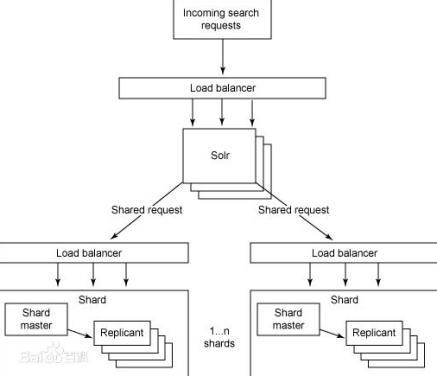


### 2.特点：

Solr是一个高性能，采用Java5开发，基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的[全文搜索引擎](https://baike.baidu.com/item/%E5%85%A8%E6%96%87%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E" \t "https://baike.baidu.com/item/Solr/_blank)。

### 3.工作方式：

文档通过Http利用XML 加到一个搜索集合中。查询该集合也是通过http收到一个XML/JSON响应来实现。它的**主要特性包括**：高效、灵活的缓存功能，垂直搜索功能，高亮显示搜索结果，通过索引复制来提高可用性，提供一套强大Data Schema来定义字段，类型和设置[文本分析](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E5%88%86%E6%9E%90" \t "https://baike.baidu.com/item/Solr/_blank)，提供基于Web的管理界面等。



### 4. Sorl里面的IK分词器的原理

Ik分词器的分词原理本质上是词典分词。现在内存中初始化一个词典，然后在分词过程中挨个读取字符，和字典中的字符相匹配，把文档中的所有的词语拆分出来的过程。

### 5. Sorl怎么设置搜索结果排名靠前（得分）？

可以设置文档中域的boost值，boost值越高，计算出来的相关度得分就越高，排名也就越靠前。此方法可以把热点商品或者推广商品的排名提高。

## 7、Zookeeper

### 1. Zookeeper是什么？

ZooKeeper是一个[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F" \t "https://baike.baidu.com/item/zookeeper/_blank)的，开放源码的[分布式应用程序](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F" \t "https://baike.baidu.com/item/zookeeper/_blank)协调服务，是[Google](https://baike.baidu.com/item/Google" \t "https://baike.baidu.com/item/zookeeper/_blank)的Chubby一个[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90" \t "https://baike.baidu.com/item/zookeeper/_blank)的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

ZooKeeper的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

ZooKeeper包含一个简单的原语集，  提供Java和C的接口。

ZooKeeper代码版本中，提供了分布式独享锁、选举、队列的接口，代码在zookeeper-3.4.3\src\recipes。其中分布锁和队列有Java和C两个版本，选举只有Java版本。

### 2. 原理：

ZooKeeper是以Fast Paxos算法为基础的，[Paxos 算法](https://baike.baidu.com/item/Paxos%20%E7%AE%97%E6%B3%95" \t "https://baike.baidu.com/item/zookeeper/_blank)存在[活锁](https://baike.baidu.com/item/%E6%B4%BB%E9%94%81" \t "https://baike.baidu.com/item/zookeeper/_blank)的问题，即当有多个proposer交错提交时，有可能互相排斥导致没有一个proposer能提交成功，而Fast Paxos作了一些优化，通过选举产生一个leader (领导者)，只有leader才能提交proposer，具体算法可见Fast Paxos。因此，要想弄懂ZooKeeper首先得对Fast Paxos有所了解。[3]

ZooKeeper的基本运转流程：

1、选举Leader。

2、同步数据。

3、选举Leader过程中算法有很多，但要达到的选举标准是一致的。

4、Leader要具有最高的执行ID，类似root权限。

5、集群中大多数的机器得到响应并follow选出的Leader。

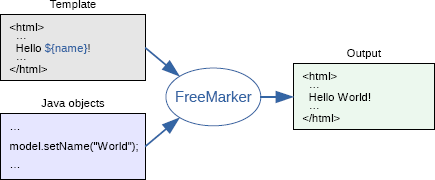
## 8、Freemarker

### 1. 什么是freemarker？

FreeMarker是一个用Java语言编写的模板引擎，它基于模板来生成文本输出。FreeMarker与Web容器无关，即在Web运行时，它并不知道Servlet或HTTP。它不仅可以用作表现层的实现技术，而且还可以用于生成XML，JSP或Java 等。

目前企业中:主要用Freemarker做静态页面或是页面展示

### 2.原理：



### 3. 使用步骤：

第一步：创建一个Configuration对象，直接new一个对象。构造方法的参数就是freemarker对于的版本号。

第二步：设置模板文件所在的路径。

第三步：设置模板文件使用的字符集。一般就是utf-8.

第四步：加载一个模板，创建一个模板对象。

第五步：创建一个模板使用的数据集，可以是pojo也可以是map。一般是Map。

第六步：创建一个Writer对象，一般创建一FileWriter对象，指定生成的文件名。

第七步：调用模板对象的process方法输出文件。

第八步：关闭流。

### 4. 网页的静态化方案

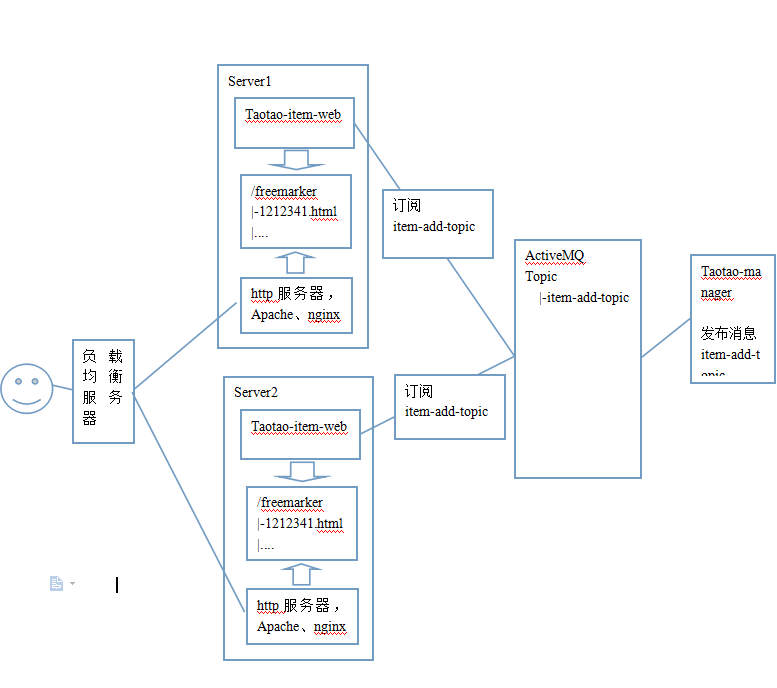
输出文件的名称：商品id+“.html”

输出文件的路径：工程外部的任意目录。

网页访问：使用nginx访问网页。在此方案下tomcat只有一个作用就是生成静态页面。

工程部署：可以把taotao-item-web部署到多个服务器上。

生成静态页面的时机：商品添加后，生成静态页面。可以使用Activemq，订阅topic（商品添加）



## 9、Nginx

### 1. Nginx是什么？

Nginx是一款高性能的http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。由俄罗斯的程序设计师Igor Sysoev所开发，官方测试nginx能够支支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

### 2. 应用场景

1. **http服务器**。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。
2. **虚拟主机**。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。
3. **反向代理，负载均衡**。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。
4. 4. Nginx 反向代理为什么可以提高网站性能？

对于后端是动态服务来说，比如 Java 和 PHP。这类服务器（如 JBoss 和 PHP-FPM）的 IO 处理能力往往不高。 Nginx 有个好处是它会把 **Request** 在读取完整之前 buffer 住，这样交给后端的就是一个完整的 HTTP 请求，从而提高后端的效率，而不是断断续续的传递（互联网上连接速度一般比较慢）。 同样，Nginx 也可以把 **response** 给 buffer 住，同样也是减轻后端的压力。

## 10、Httpclient

### 1. Httpclient 是什么？

HttpClient不是一个浏览器。它是一个客户端的HTTP通信实现库。HttpClient的目标是发送和接收HTTP报 文。HttpClient不会去缓存内容，执行嵌入在HTML页面中的javascript代码，猜测内容类型，重新格式化 请求/重定向URI，或者其它和HTTP运输无关的功能.它主要就是支持HTTP传输协议的.

### 2. Httpclient 的使用

我们知道，HTTP协议的连接方法有GET、POST、PUT和HEAD方式，在创建Method实例的时候可以更具具体的 方法来创建。HttpClient的使用一般分如下几步：

1. 创建HttpClient实例。

2、创建具体连接方法的实例。如POST方法创建PostMethod的实例，在实例化时从构造函数中传 入待连接的URL地址。

3、对post的发送内容等信息进行配置

4、执行HttpClient的execute方法

5、如果返回的状态码正常，表明连接成功，可以读取response的内容

## 11、Jsonp

### 1. jsonp 到底是什么？

JSONP 的原理非常简单，**为了克服跨域问题，利用没有跨域限制的 script 标签加载预设的 callback 将 内容传递给 js。**一般来说我们约定通过一个参数来告诉服务器 JSONP 返回时应该调用的回调函数名，然 后拼接出对应的 js。

## 12、Mycat中间件

《MyCat》是代替昂贵的oracle的MySQL集群中间件。

### 1. 为什么需要MyCat？

虽然云计算时代，传统数据库存在着先天性的弊端，但是NoSQL数据库又无法将其替代。如果传统数据易于扩展，可切分，就可以避免单机（单库）的性能缺陷。

MyCat的目标就是：低成本地将现有的单机数据库和应用平滑迁移到“云”端，解决数据存储和业务规模迅速增长情况下的数据瓶颈问题。2014年MyCat首次在上海的《中华架构师》大会上对外宣讲引发围观，更多的人参与进来，随后越来越多的项目采用了MyCat。

### 2. MyCat是什么？

从定义和分类来看，它是一个开源的分布式数据库系统，是一个实现了MySQL协议的服务器，前端用户可以把它看作是一个数据库代理，用MySQL客户端工具和命令行访问，而其后端可以用MySQL原生协议与多个MySQL服务器通信，也可以用JDBC协议与大多数主流数据库服务器通信，其核心功能是分表分库，即将一个大表水平分割为N个小表，存储在后端MySQL服务器里或者其他数据库里。

MyCat发展到目前的版本，已经不是一个单纯的MySQL代理了，它的后端可以支持MySQL、SQL Server、Oracle、DB2、PostgreSQL等主流数据库，也支持MongoDB这种新型NoSQL方式的存储，未来还会支持更多类型的存储。而在最终用户看来，无论是那种存储方式，在MyCat里，都是一个传统的数据库表，支持标准的SQL语句进行数据的操作，这样一来，对前端业务系统来说，可以大幅降低开发难度，提升开发速度

## 13、Git

Git是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

### Git 与 SVN 区别

GIT不仅仅是个版本控制系统，它也是个内容管理系统(CMS),工作管理系统等。

如果你是一个具有使用SVN背景的人，你需要做一定的思想转换，来适应GIT提供的一些概念和特征。

Git 与 SVN 区别点：

* 1、GIT是分布式的，SVN不是：这是GIT和其它非分布式的版本控制系统，例如SVN，CVS等，最核心的区别。
* 2、GIT把内容按元数据方式存储，而SVN是按文件：所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs等的文件夹里。
* 3、GIT分支和SVN的分支不同：分支在SVN中一点不特别，就是版本库中的另外的一个目录。
* 4、GIT没有一个全局的版本号，而SVN有：目前为止这是跟SVN相比GIT缺少的最大的一个特征。
* 5、GIT的内容完整性要优于SVN：GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏