目录

[1、 构件和连接件 2](#_Toc514751158)

[2、 软件体系结构生命周期模型 3](#_Toc514751159)

[3、软件重用技术在软件开发中的作用？ 3](#_Toc514751160)

[4、软件体系结构的生命周期模型和软件生命周期模型有什么关系? （补充） 3](#_Toc514751161)

[6、C2 概述 4](#_Toc514751162)

[13、HDFS 处理过程 5](#_Toc514751163)

[14、MapReduce 处理过程 6](#_Toc514751164)

[15、MapReduce分布式处理技术-实现机制 7](#_Toc514751165)

[16、MapReduce分布式处理技术-实例-单词统计WordCount 8](#_Toc514751166)

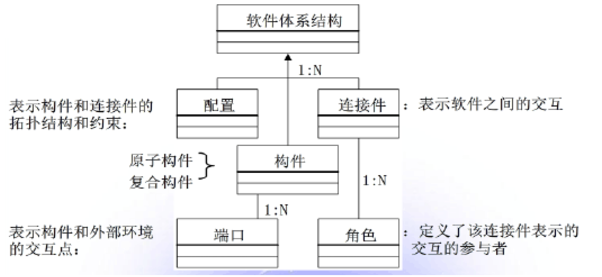
[17、MapReduce分布式处理技术-实例-文档倒排索引算法 10](#_Toc514751167)

[19、中间件的优点 11](#_Toc514751168)

[20、架构设计的基本准则 11](#_Toc514751169)

## 构件和连接件

软件体系结构的核心模型由五种元素组成：构件、连接件、配置、端口、角色。其中，构件、连接件和配置是最基本的元素。



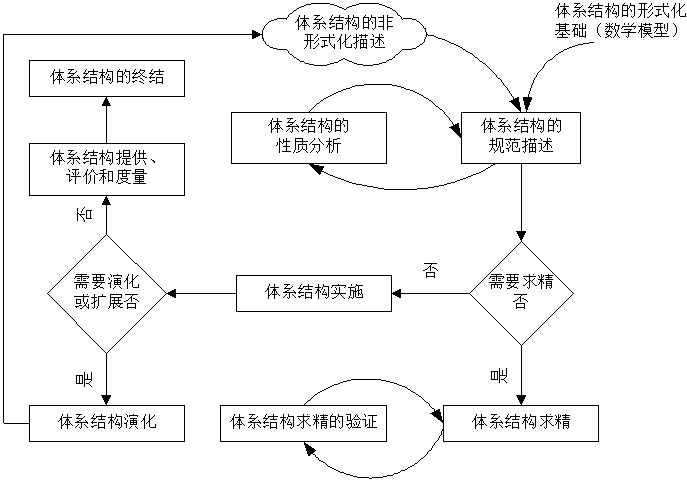
**构件：**具有某种功能的可重用软件单元，表示系统中主要的计算和数据存储。构件只能通过接口与外部交互，接口由一组端口组成，每个端口表示了构件与外部环境的交互点。通过不同的端口类型，一个构件可以提供多重接口。

（每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。）

构件的定义：构件是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统；结构上，它是语义描述、通讯接口和实现代码的复合体。

**连接件：**表示了构件间的交互；连接件也有接口，由一组角色组成，每个角色定义了该连接件表示的交互的参与者。

## 软件体系结构生命周期模型



**模型解释：**

1、软件体系结构的非形式化描述;

2、软件体系结构规范描述和分析;

3、软件体系结构的求精及其验证;

4、软件体系结构的实施 ;

5、软件体系结构的演化和扩展;

6、软件体系结构提供、评价和度量;

7、软件体系结构的终结

## 3、软件重用技术在软件开发中的作用？

软件重用可以分为：代码重用，设计结果重用，分析结果的重用，测试信息的重用，体系结构的重用。

提高软件生产率，降低软件开发的成本； 提高软件质量； 互操作性好； 支持原型开发；

提高了软件的可维护性、可扩展性、可理解性。 举例：1976年IBM、HP、NEC、AT&T的调查显示：

基于软件重用的软件产品线的发展使开发时间缩短了1.5—2倍，维护成本降低了2-5倍，软件质量提高了5—10倍，开发成本降低了12%—15%

## 4、软件体系结构的生命周期模型和软件生命周期模型有什么关系? （补充）

软件生命周期 同任何事物一样,一个软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生存周期（软件生命周期）。

软件生命周期(SDLC，软件生存周期)是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。 软件生命周期模型 通俗说，就是软件开发过程中所遵循的模式。它运用了结构化的设计思想；具有严格的顺序性和依赖性，必须等待前一阶段的工作完成之后，才能开始后一阶段的工作，并且其一阶段的输出就是后一阶段的输入。具体有： 瀑布（waterfall）模型、原型（prototyping）模型、增量（incremental）模型、螺旋（spiral）模型、快速应用开发（RAD）模型、渐进式模型等。

软件体系结构生命周期模型是对软件生命周期模型的重用，是软件生命周期模型的某一阶段的细化。它也继承了结构化设计思想，运用了面向过程的设计方法。它也具有顺序性和依赖性。不过它还添加了一些跳转，当某一阶段比较细致或软件的结构比较简单时可以跳过某一阶段而经如下一阶段。软件体系结构生命周期模型包括非形式化的描述；体系结构的规范描述和分析；体系结构的求精及其验证；软件体系结构的事实；软件体系结构的演化和扩展；软件体系结构的提供、评价和度量；软件体系结构的终结。

## 6、C2 概述

◎ C2和其提供的设计环境（Argo）支持采用基于时间的风格来描述用户界面系统，并支持使用可替换、可重用的构件开发GUI的体系结构。

◎ 在C2中，连接件负责构件之间消息的传递，而构件维持状态、执行操作并通过两个名字分别为“top”和“bottom”的端口和其它的构件交换信息。

◎ 每个接口包含一种可发送的消息和一组可接收的消息。构件之间的消息要么是请求其它构件执行某个操作的请求消息，要么是通知其他构件自身执行了某个操作或状态发生改变的通知消息。

◎ 构件之间的消息交换不能直接进行，而只能通过连接件来完成。每个构件接口最多只能和一个连接件相连，而连接件可以和任意数目的构件或连接件相连。

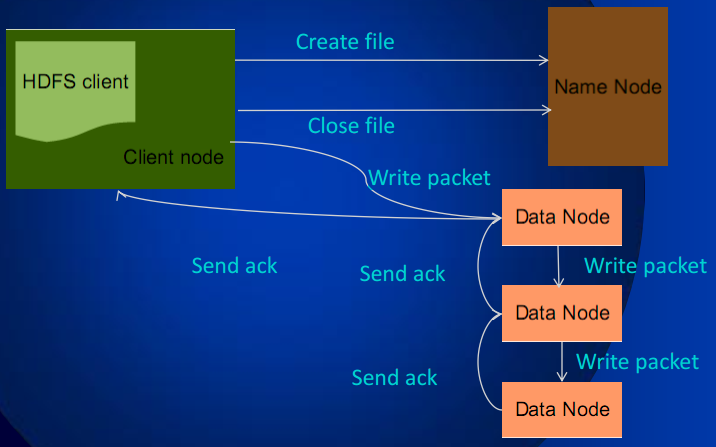
◎ 请求消息只能向上层传送而通知消息只能向下层传送。

◎ 通知消息的传递只对应于构件内部的操作，而和接收消息的构件的需求无关。

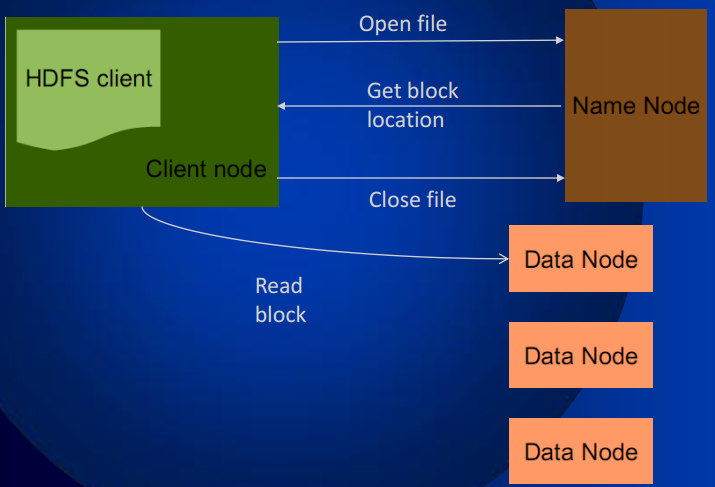
◎ C2对构件和连接件的实现语言、实现构件的线程控制、构件的部署以及连接件使用的通讯协议等都不加限制。

## 13、HDFS 处理过程

（1）HDFS如何写文件：



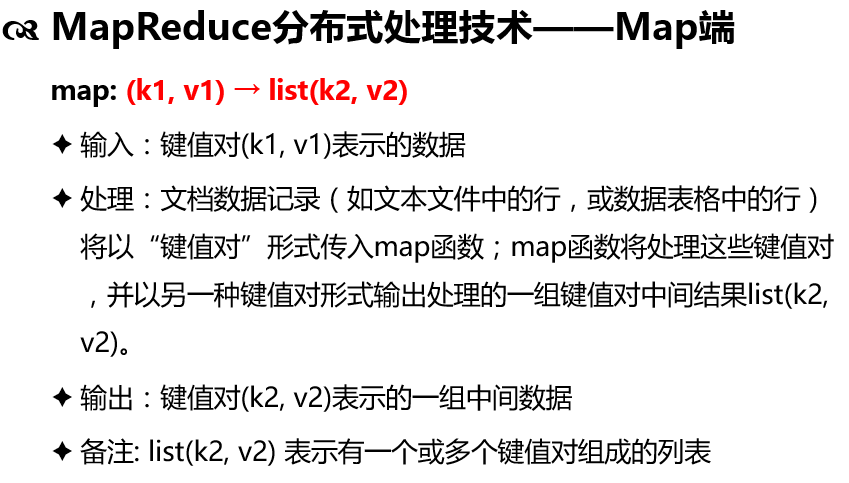
（2）HDFS如何读文件：

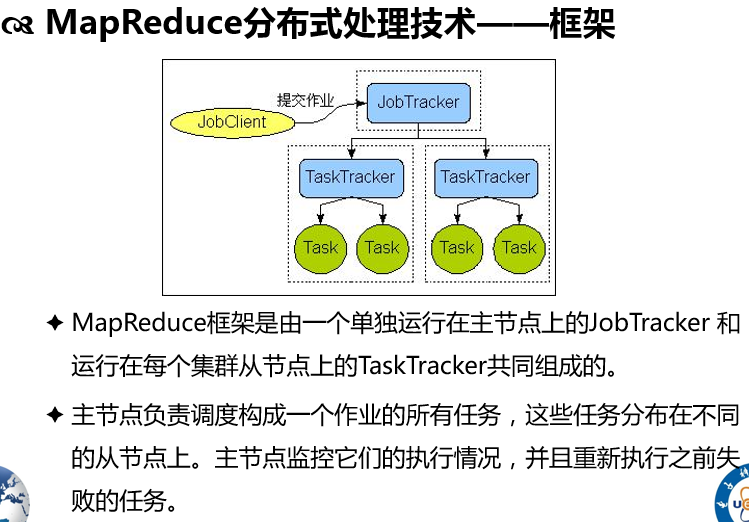
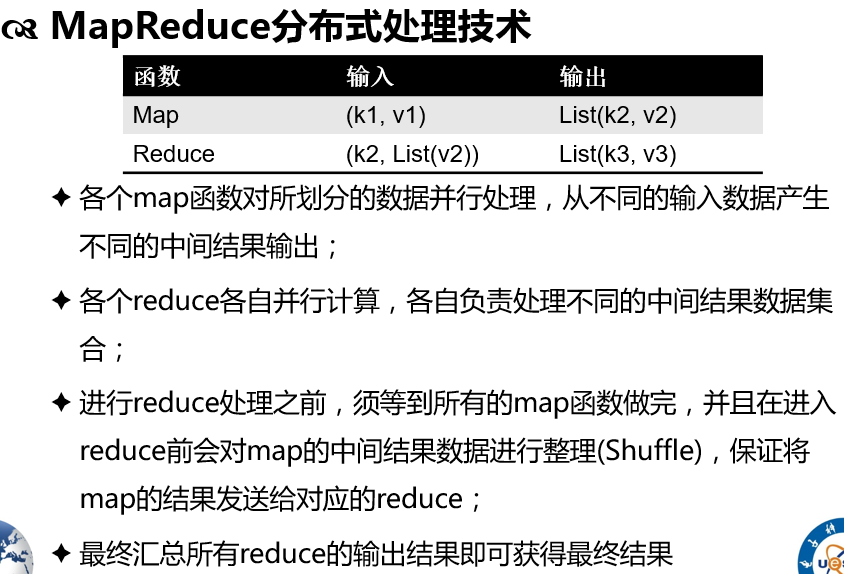
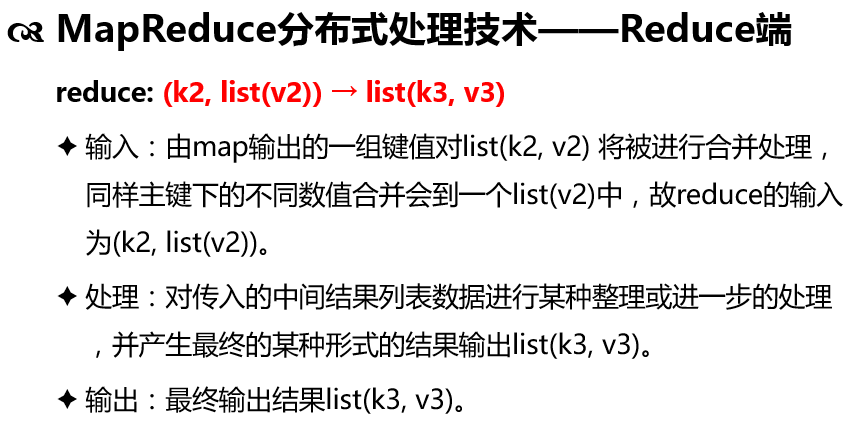


（3）HDFS上传文件流程：

当客户端上传数据时，先向NamaNode请求创建文件，然后NameNode会返回存储的DataNode节点和位置信息。随后客户端先通过管道将文件传到本地硬盘，每凑满指定大小（默认是64M），再一起上传到一个DataNode， DataNode收到文件后会返回确认信息，并以4K为单位传到下一DataNode。该文件上传方式被称为“流水式”。

## 14、MapReduce 处理过程





## 15、MapReduce分布式处理技术-实现机制

(1) MapReduce函数库首先把输入文件分成M块，每块大概16MB到64MB。接着在集群的机器上执行处理程序。

* MapReduce算法运行过程中有一个主控程序，称为master。主控程序会产生很多作业程序，称为worker。并且把M个map任务和R个reduce任务分配给这些worker，让它们去完成。

(2) 被分配了map任务的worker读取并处理相关的输入(这里的输入是指已经被切割的输入小块splite)。它处理输入的数据，并且将分析出的键/值(key/value)对传递给用户定义的reduce()函数。map()函数产生的中间结果键/值(key/value)对暂时缓冲到内存。

(3) map()函数缓冲到内存的中间结果将被定时刷写到本地硬盘，这些数据通过分区函数分成R个区。这些中间结果在本地硬盘的位置信息将被发送回master，然后这个master负责把这些位置信息传送给reduce()函数的worker。

(4) 当master通知了reduce()函数的worker关于中间键/值(key/value)对的位置时，worker调用远程方法从map()函数的worker机器的本地硬盘上读取缓冲的中间数据。当reduce()函数的worker读取到了所有的中间数据，它就使用这些中间数据的键(key)进行排序，这样可以使得相同键(key)的值都在一起。

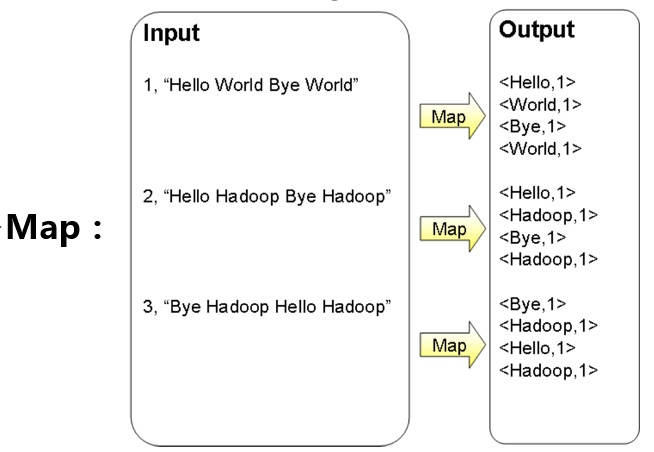
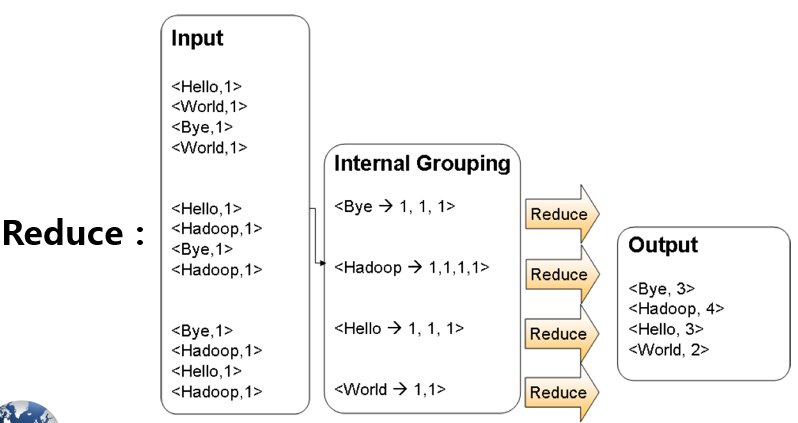
(5) reduce()函数的worker根据每一个中间结果的键(key)来遍历排序后的数据，并且把键(key)和相关的中间结果值(value)集合传递给reduce()函数。reduce()函数的worker最终把输出结果存放在master机器的一个输出文件中。

(6) 当所有的map任务和reduce任务都已经完成后，master激活用户程序。在这时，MapReduce返回用户程序的调用点。

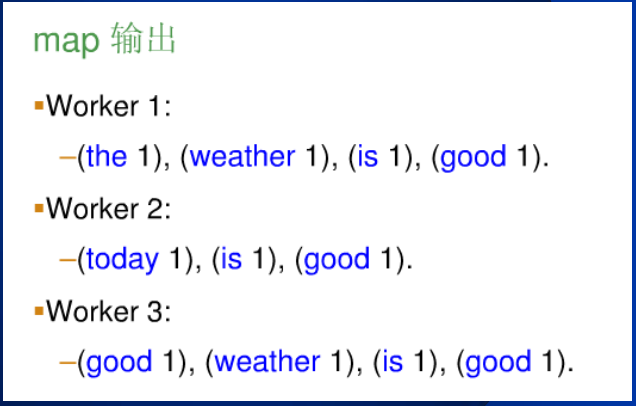
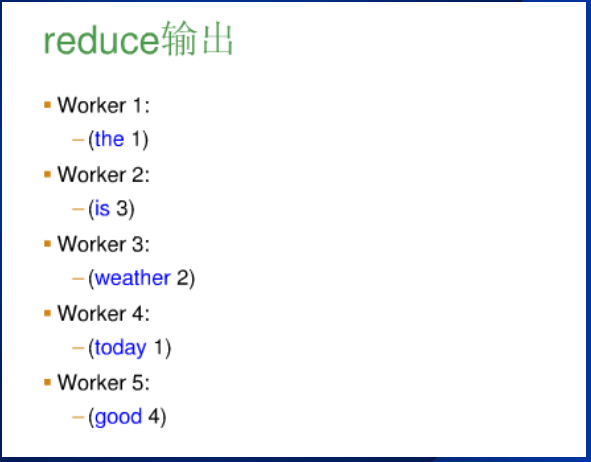
(7) 当以上步骤成功结束以后，MapReduce的执行数据存放在总计R个输出文件中(每个输出文件都是由reduce任务产生的，这些文件名是用户指定的)。通常，用户不需要将这R个输出文件合并到一个文件，他们通常把这些文件作为输入传递给另一个MapReduce调用，或者用另一个分布式应用来处理这些文件，并且这些分布式应用把这些文件看成为输入文件由于分区(partition)成为的多个块文件。

## 16、MapReduce分布式处理技术-实例-单词统计WordCount

* 问题描述：统计文本中所出现单词的次数。
* 解决思路：需符合Map、Reduce各自的输入、输出格。
* Map端：输入为(k1, v1), 以文本行号为k1，以行内容为v1；输出为list(k2, v2)，每有一个单词就输出一个(word, 1)。
* Reduce端：输入为(k2, list(v2))，将list(v2)中所有数字相加即可得到单词次数；输出为list(k3, v3)，即最终的结果：(单词, 单词次数)。

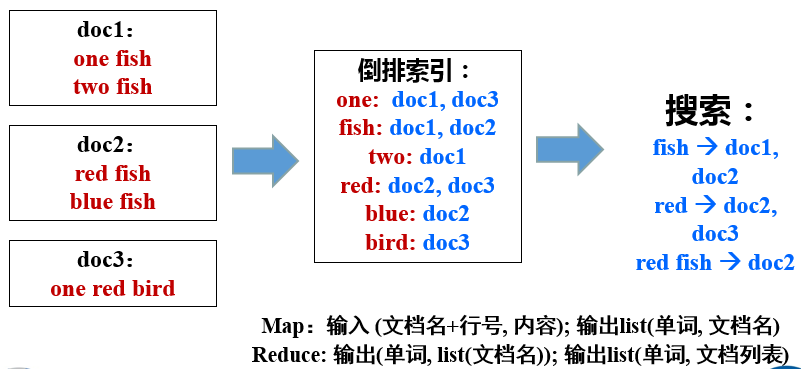
 

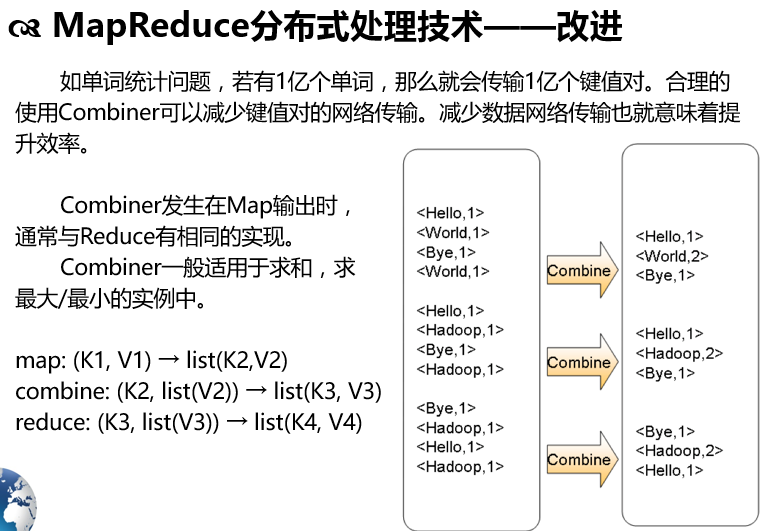
**以下是书上的案例：**

## 17、MapReduce分布式处理技术-实例-文档倒排索引算法

* 问题描述：给出气象站历年每天的数据，要提取出每年的最高气温。数据示例： 0067011990999991950051507004…..99999N9+00221+99999…
* 解决思路-Map端： 输入为(k1, v1): (行号，内容)；输出为list(k2, v2): (年份，温度)，例如(1950, 22)。
* 解决思路-Reduce端： 输入为(k2, list(v2): 如(1950, [22, -11])，对list(v2)排序可得到高气温；输出为list(k3, v3)，即最终的结果：(年份, 最高气温)。
* 问题描述：Inverted Index(倒排索引)是目前几乎所有支持全文检索的搜索引擎都要依赖的一个数据结构。基于索引结构，给出一个词(term)，能取得含有这个term的文档列表(the list of documents)





## 19、中间件的优点

1、缩短应用开发周期；2、减少项目开发的风险，已经过大量测试完善；3、应用系统质量的可维护性；4、增加产品的吸引力；5、透明与其它应用程序的交互；6、与运行平台提供网络通信服务无关；7、具有良好的可靠性和可用性；8、良好的可扩展性；

## 20、架构设计的基本准则

1、抽象；2、封装；3、信息隐藏；4、模块化；5、注意点分离；6、内聚与耦合

**如何实现需求端到服务器端两者之间的对接：**

答：以查询名录的行为方式去找到服务。

2 、中介代理的作用：连接需求端到服务器端的桥梁。

1. 中介在架构中的作用：客户和服务器之间无需直接了解；客户和服务器不必一对一；

客户可以在运行的时候定位服务并交互