大型电商分布式系统实践 第1周

DATAGURU专业数据分析社区

什么样的网站才能够称得上是大型网站

访问量大的网站就是大型网站？

什么样的网站才能够称得上是大型网站

站点 hao123.com 的全球网站排名查询结果

当日排名	排名变化趋势	一周平均排名	排名变化趋势	一月平均排名	排名变化趋势	三月平均排名	排名变化趋势
15	↑4	19	↓1	16	↓2	15	0

访问量大的网站就是大型网站，这个说法不全对，从Alexa全球网站排名来看，hao123排名15，但是，hao123的主要内容就只有一张页面，从技术实现上来说，应该非常简单。

因此，在我看来，光关注访问量还不够，我们还得关注另外一个维度的条件，即一个大型网站应该要有海量的数据，访问量和数据二者缺一不可。

正是由于大型网站大访问量和大数据量的两个属性，给技术人员带来了一系列的挑战。

大型分布式网站的架构演进 ------单一应用架构

从一个简单的电商网站说起，它可能包含如下的几个模块和功能，如首页、detail页、list页、下单页、支付页以及后台管理等页面和功能。

大型分布式网站的架构演进 ------垂直应用架构

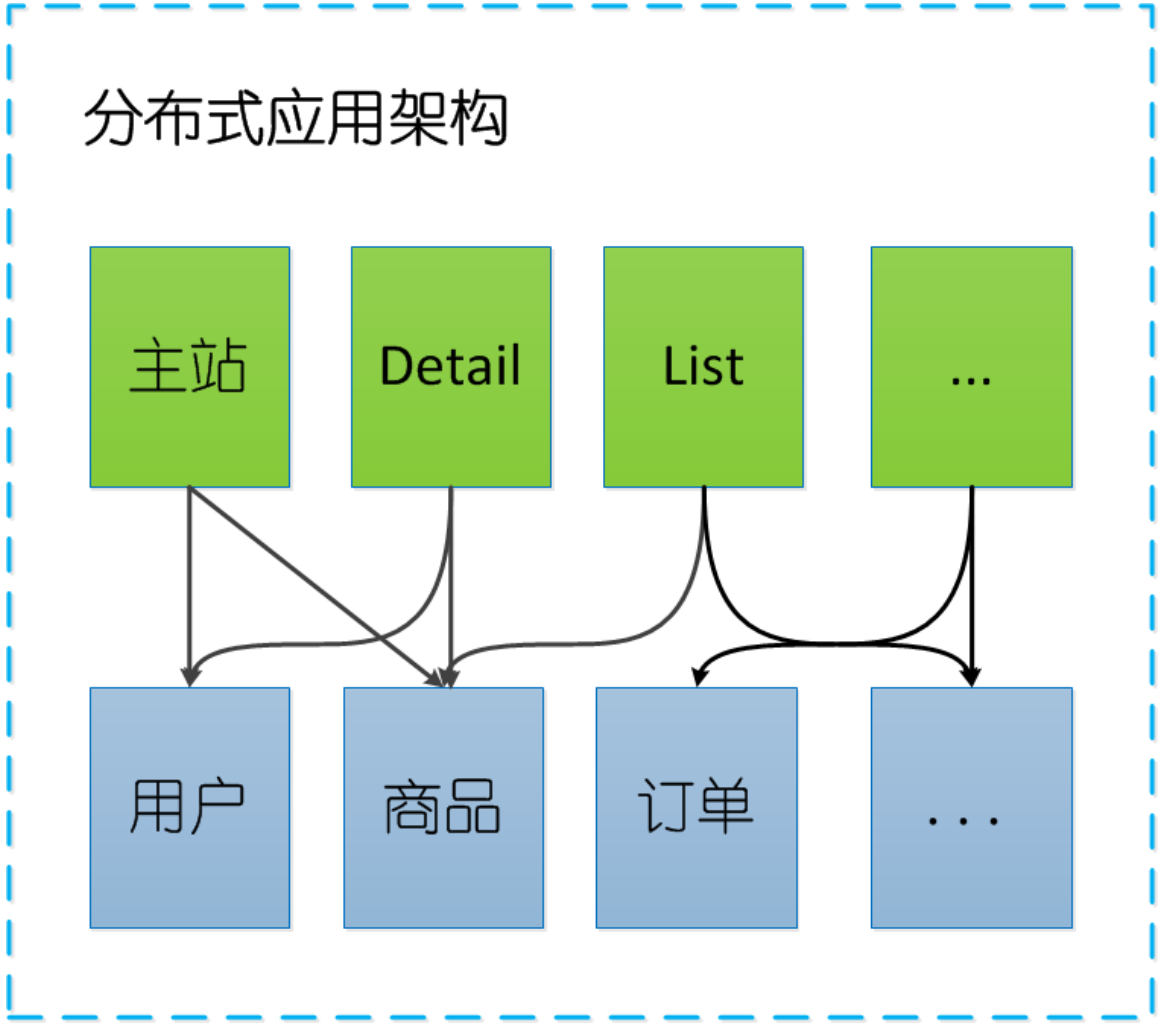
随着业务的发展，单一应用架构带来的问题是：
1. 代码越来庞大，业务越来越复杂，多个团队开发同一个应用，难以维护
2. 业务复杂，占用的系统资源越来越多，流量越来越大，不方便扩展

垂直应用架构解决了单一应用架构所面临的扩容问题，流量能够分散到各个子系统当中，且系统的体积可控，一定程度上降低了开发人员之间协同以及维护的成本，提升了开发效率。

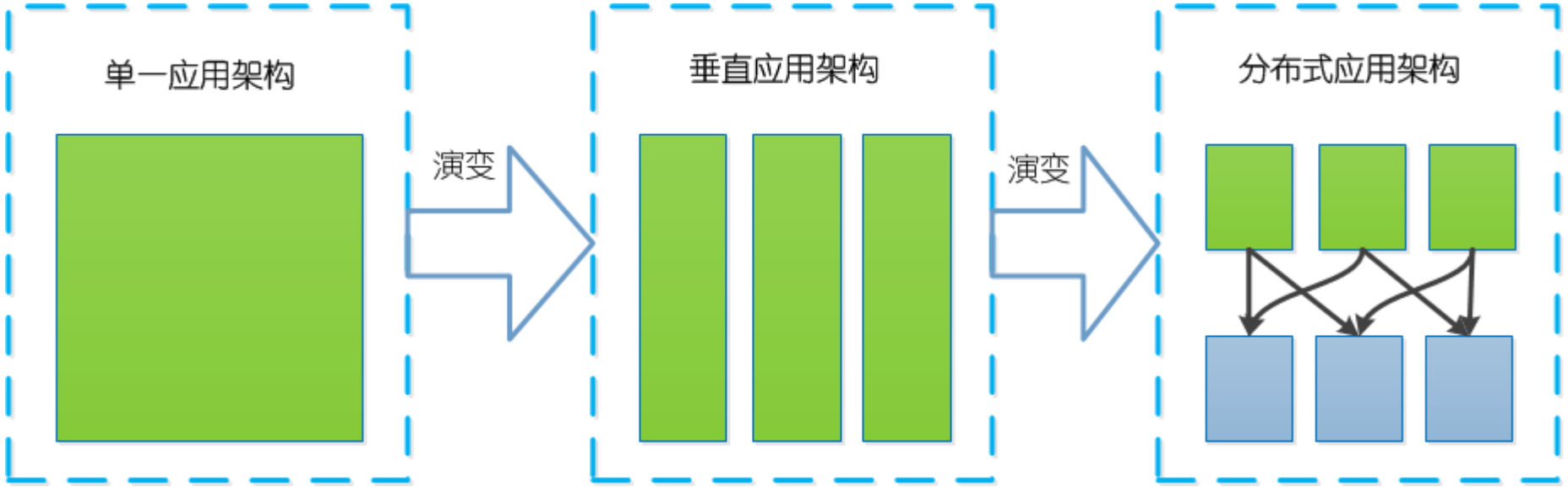
大型分布式网站的架构演进------分布式应用架构

垂直应用体系带来的缺陷是，相同逻辑的代码在不同的垂直应用中复制，不能复用，难以维护和升级。

将公共的业务逻辑提取出来，形成服务，对外提供，避免重复造轮子，相同的逻辑维护一份，也便于升级改造，原本需要一个大团队维护的系统，也可以切分成一个个子系统，分配给一个个固定小团队来维护，降低了系统发布的风险，提高了系统的稳定性，并且也可以让前端业务系统与底层数据访问分离，团队分工更为明确。



大型分布式网站的架构演进



随着单一应用架构、垂直应用架构向分布式应用架构的演变，业务规模越来越庞大，系统逻辑越来越复杂，系统研发和维护团队规模也越来越大，又发展成为面向服务的架构体系(SOA)，并且，部分通用的功能和组件抽离出来，形成一系列中间件，加上一系列的分布式系统的基础设施，共同组成了大型分布式网站的复杂架构。

分布式系统的中间件及基础设施

在系统逐步服务化的同时，一个大型、稳健、成熟的分布式系统背后，往往会涉及众多的中间件，以及一系列的支撑系统，我们将这些中间件和支撑系统称为分布式系统的基础设施。

这些分布式系统所依赖的基础设施包括**服务框架、消息中间件、数据访问中间件、配置中心、分布式缓存系统、持久化存储(关系数据库、nosql数据库)、搜索引擎、CDN网络、负载均衡系统、运维自动化系统、硬件虚拟化及镜像管理系统、分布式文件系统、日志收集系统、监控系统、离线计算、实时计算、数据仓库**等等。

面向服务的体系架构------SOA

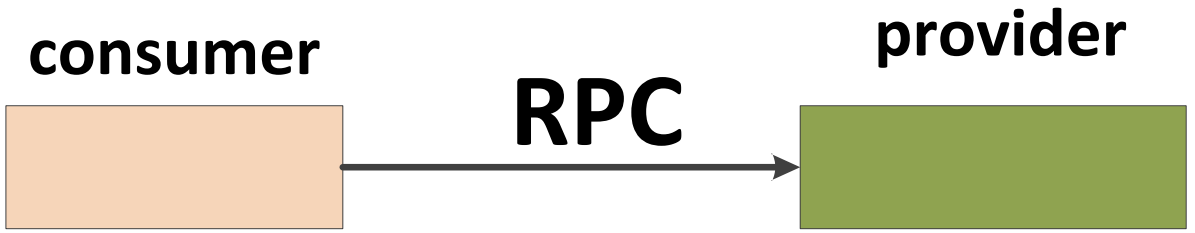
当系统达到一定规模，不同系统之间存在着重叠的业务，容易形成信息孤岛，重复造轮子，这种情况下，应用之间相互交互、相互调用便不可避免。此时，相对核心的业务将会被抽取出来，作为单独的系统对外提供服务，达成业务之间相互复用。

互联网企业崇尚一个快字，上层业务都想借用已有的底层服务，来快速搭建更多更丰富的应用，降低新业务开展的人力和时间成本，快速满足瞬息万变的市场需求。因此，对于业务逻辑复用的需求十分强烈，公共的业务被拆分出来，形成可共用的服务，最大程度的保障了代码和逻辑的复用，避免重复建设，这便是服务化的初衷。

随着服务化的进一步发展，服务越来越多，服务之间的调用和依赖关系也越来越复杂，诞生了面向服务的架构体系(SOA)，也因此衍生出了一系列相应的技术，如对服务提供、服务调用、连接处理、通信协议、序列化方式、服务发现、服务路由、日志输出等行为进行封装的服务框架，以及为大规模的服务化应用保驾护航的服务治理系统。

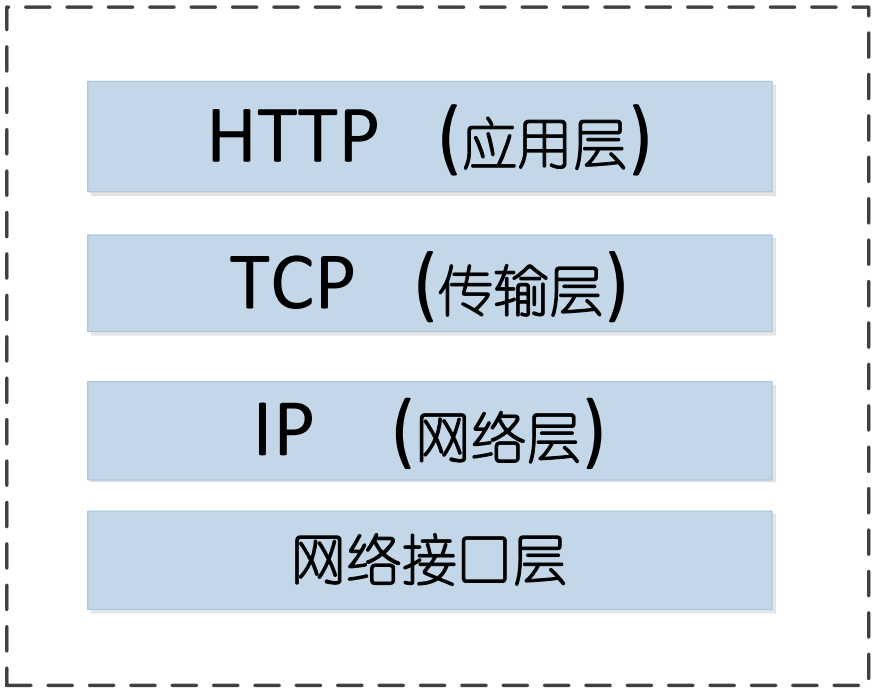
RPC远程调用

RPC的全称是Remote Process Call，即远程过程调用，它应用广泛，实现方式也很多，拥有包括RMI、WebService等等诸多成熟的方案，在业界得到了广泛的使用。



RPC的实现包括客户端和服务端，即服务的调用方以及服务的提供方，服务调用方发送RPC请求到服务提供方，服务提供方根据调用方提供的参数执行请求方法，将执行结果返回给调用方，一次RPC调用完成。

通信协议---协议栈

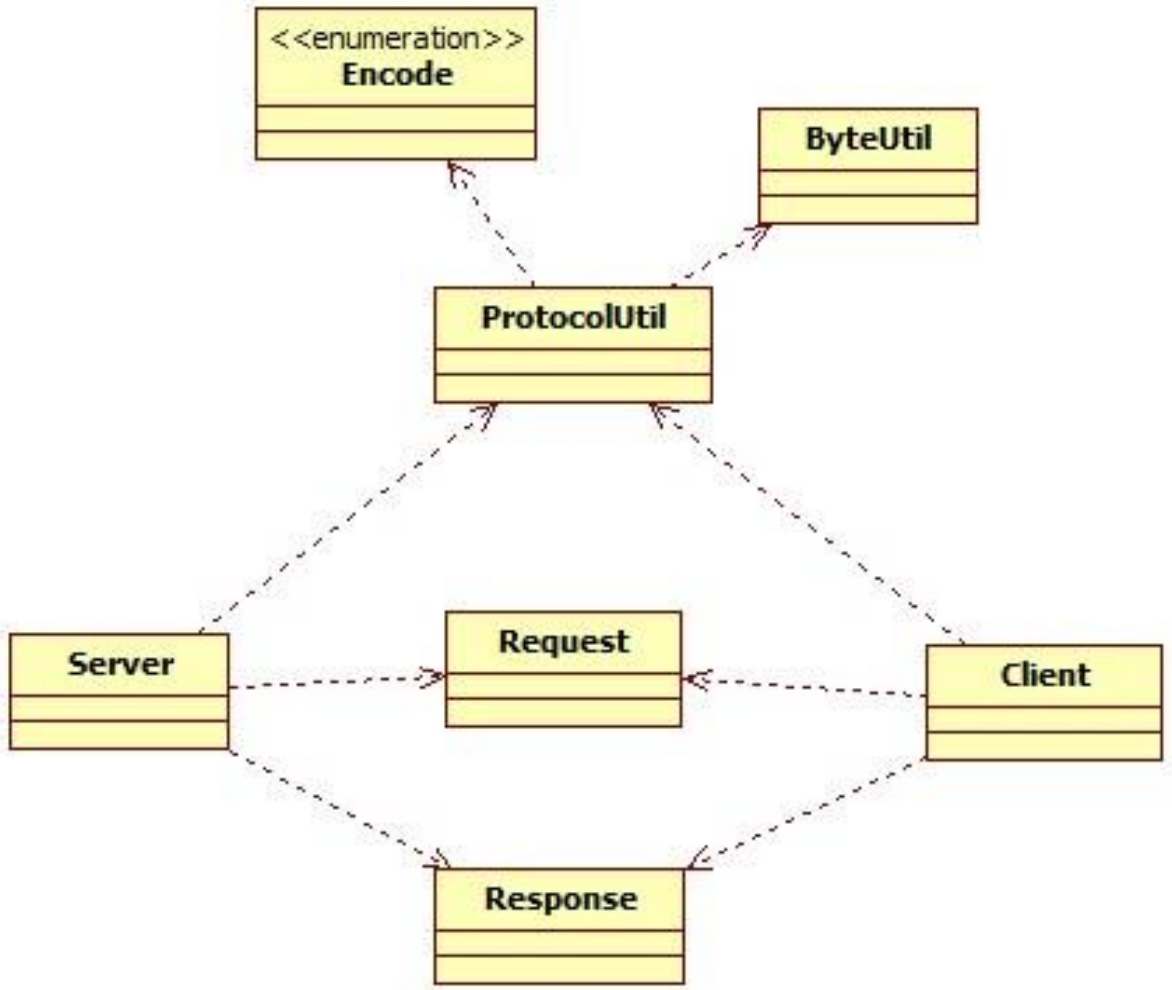


HTTP网络协议栈

协议是通信的规范，根据TCP/IP协议模型，HTTP协议属于应用层协议，它构建在TCP和IP协议之上，处于TCP/IP体系架构中的最顶端，这样一来，它便不需要处理下层协议间诸如丢包补发，握手以及数据的分段和重新组装等等繁琐的细节，从而使开发人员可以专注于上层应用的设计。

为了更好的理解协议栈，我们可以基于java的Socket API接口，通过设计一个简单的应用层通信协议，来窥探协议实现的一些过程与细节。

通信协议---一个简单协议的实现



RPC通信协议的选择-TCP协议

基于TCP协议实现的RPC，由于处于协议栈的下层，能够更灵活的对协议字段进行定制，减少网络传输字节数，降低网络开销，提高性能，达到更大的吞吐量和并发数，但是需要更多的关注的底层复杂细节，实现的代价更高，较难实现跨平台的调用。

而随着请求规模的扩展，基于TCP协议RPC的实现，程序需要考虑多线程并发，锁，IO等等复杂的底层细节的实现，实现起来较为复杂。在大流量高并发压力下，任意一个细小的错误，都会被无限放大，最终导致程序宕机。

RPC通信协议的选择—HTTP协议

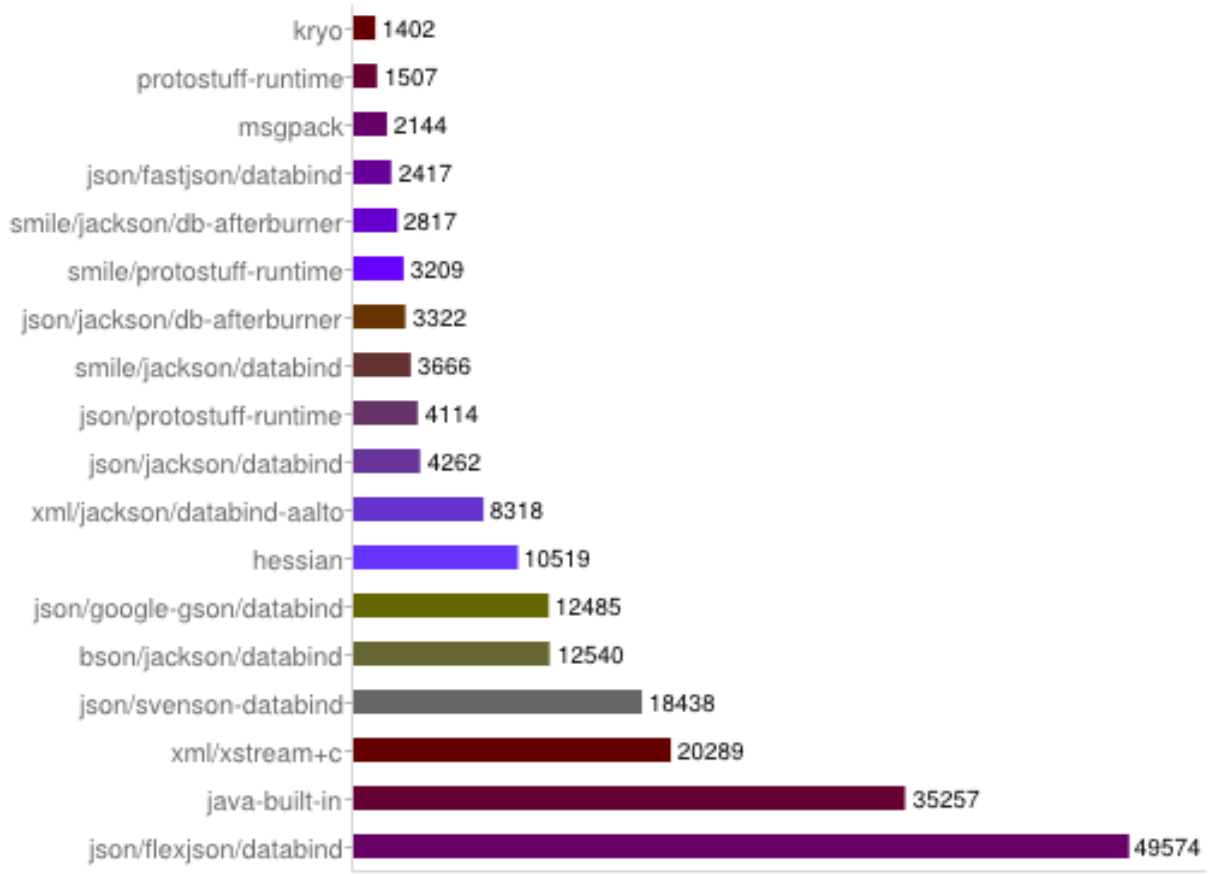
基于HTTP协议的RPC，可以使用JSON或者是XML格式的响应数据，而JSON和XML作为通用的格式标准，开源的解析工具已经相当成熟，在其上进行二次开发屏蔽了很多底层繁琐的细节，非常便捷和简单。
而对于基于HTTP协议的实现来说，很多成熟的开源WEB容器已经帮其处理好这些事情，如tomcat、jboss、apache等等，开发人员可以将更多的精力集中在业务的实现上，而非底层细节的处理。

当然，基于HTTP协议的实现，也有其处于劣势的一面。由于是上层协议，发送包含同等内容的信息，使用HTTP协议传输所占用的字节数肯定要比使用TCP协议传输所占用的字节数更多。因此，同等网络环境下，通过HTTP协议传输相同内容，效率会比基于TCP协议的数据传输要低，信息传输所占用的时间要更长。
当然，通过优化代码实现以及使用gzip数据压缩，能够缩小这一差距。通过权衡利弊，结合实际环境中其性能对于用户体验的影响来看，基于HTTP协议的RPC还是有很大优势的。

对象序列化方式的选择

什么是对象序列化：

将对象转换为能够在网络上传输的二进制流的过程称为**对象的序列化**。
将二进制流恢复为对象的过程称为**对象的反序列化**。



序列化的性能对比

序列化方式的选择

Google的Protocol Buffers真正开源出来的时间并不长，但是其性能优异，短时间内引起了广泛的关注。其优势是性能十分优异，支持跨平台，但使用其编程代码侵入性较强，需要编写proto文件，无法直接使用Java等面向对象编程语言的对象。相对于Protocol Buffers，Hessian的效率稍低，但是其对各种编程语言有着良好的支持，且性能稳定，比Java本身内置的序列化方式的效率要高很多。Java内置的序列化方式不需要引入第三方包，使用简单，在对效率要求不是很敏感的场景下，也未尝不是一个好的选择。而的Xml和Json格式，在互联网领域尤其是现在流行的移动互联网领域，得益于其跨平台的特性，得到了极为广泛的应用。

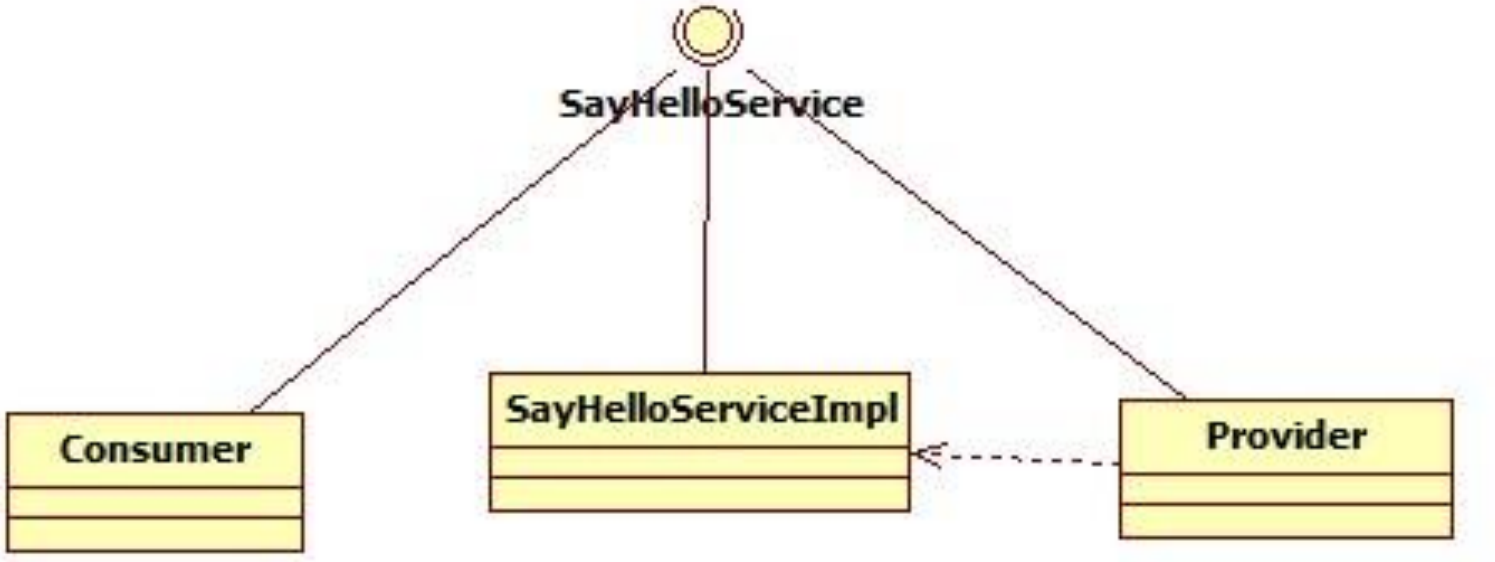
基于java的序列化方式

基于hession的序列化方式

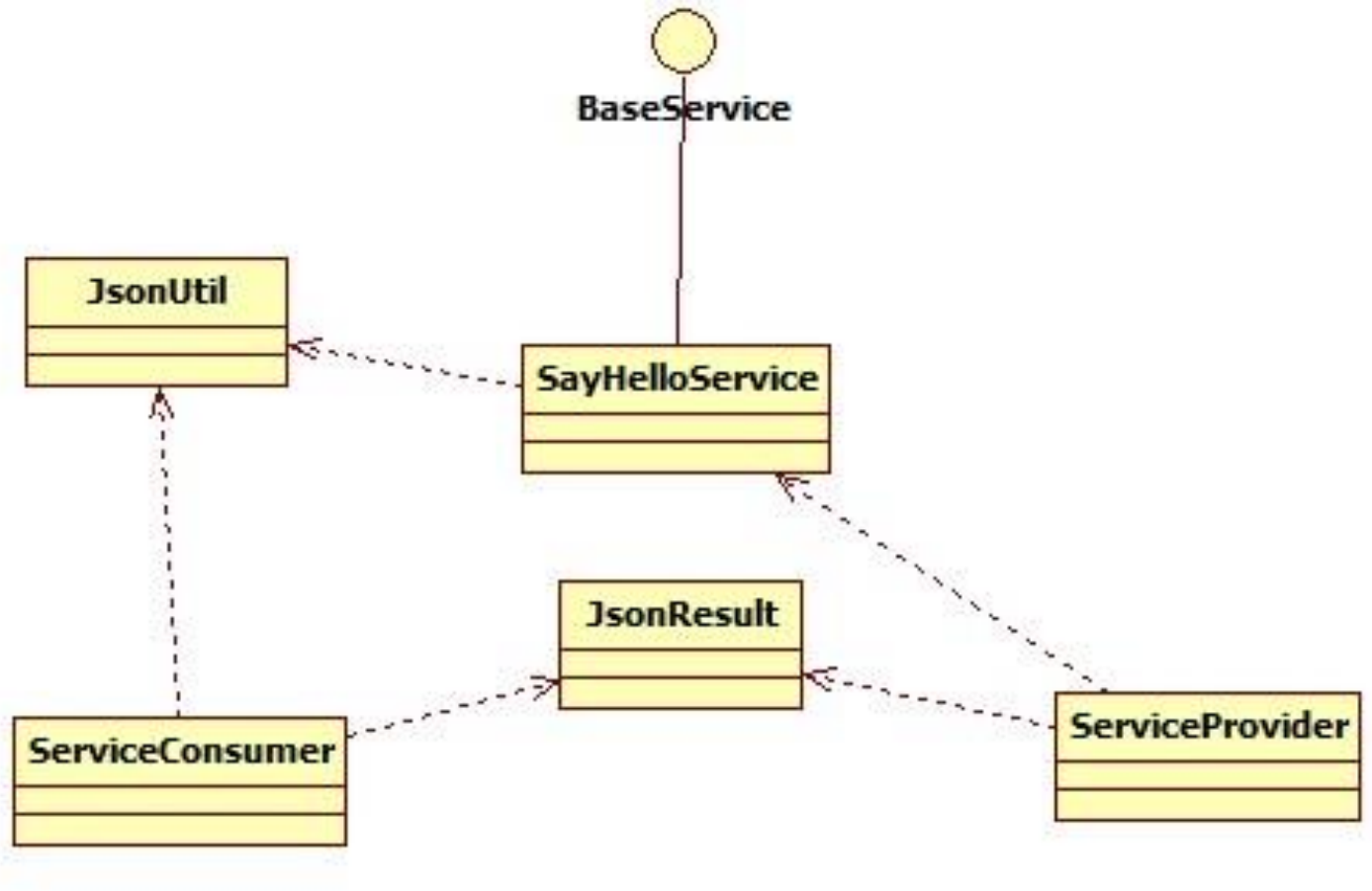
基于JSON的序列化方式

基于XML的序列化方式

远程调用的实现 – 基于TCP的远程调用



远程调用的实现—— 基于HTTP的远程调用



两种URL风格— RPC

RPC风格的URL比较好理解，直接在HTTP请求的参数中标明需要远程调用的服务接口名称，服务需要的参数，如下所示：
http://hostname/provider.do?service=com.http.sayhello&format=json&timestamp=2013-07-07-13-22-09&arg1=arg1&arg2=arg2

hostname表示服务提供方的主机名，service表示远程调用的服务接口名称，format表示返回参数的格式，timestamp表示客户端请求的时间戳，arg1和arg2表示服务所需要的参数。

两种URL风格—RESTful

POST	http://hostname/people	创建name为zhangsan的people记录
GET	http://hostname/people/zhangsan	返回name为zhangsan的people记录
PUT	http://hostname/people/zhangsan	提交name为zhangsan的people记录更新
DELETE	http://hostname/people/zhangsan	删除name为zhangsan的people记录

RESTful风格其中的一个思想是，通过HTTP请求对应的POST、GET、PUT、DELETE方法，来完成对应的CRUD 操作。

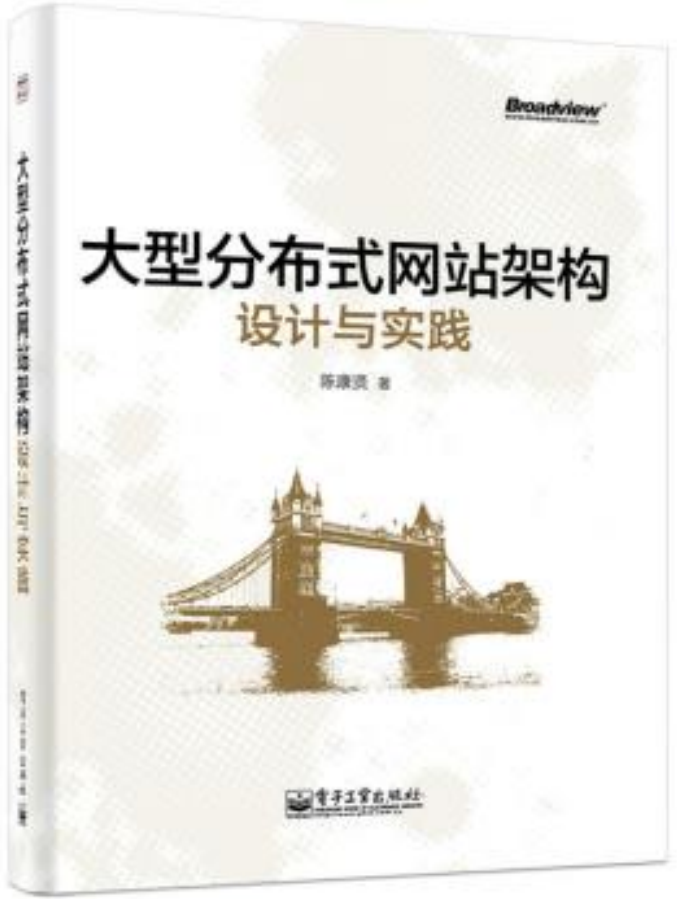
两种URL风格—RESTful 结合 RPC

POST arg1=hello arg2=123

URL http://hostname/provider/sayhelloservice/2013-07-07-13-22-09.json

URL中hostname表示的是服务提供方的主机名，provider表示访问的是服务提供方，sayhelloservice是对应的服务接口名称，.json表示的是需要服务端返回的数据格式，2013-07-07-13-22-09表示的是客户端访问的时间戳，arg1和arg2参数采用POST方式发送到服务端。

推荐一本书



问题

1. 大家理解的大型网站是怎样的

2. 大家对架构师这个岗位的理解是？

Thanks

FAQ时间