> 慕课网慕课教程 9. rpc、http 以及 restful 之间的区涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

你一定会觉得这个问题很奇怪，是的，包括我，但是你在网络上一搜，会发现类似对比的文章比比皆是，我在想可能很多初学者由于基础不牢固，才会将不相干的二者拿出来对比吧。既然是这样，那为了让你更加了解陌生的 RPC，就从你熟悉得不能再熟悉的 REST 入手吧。

REST，是 Representational State Transfer 的简写，中文描述表述性状态传递（是指某个瞬间状态的资源数据的快照，包括资源数据的内容、表述格式 (XML、JSON) 等信息。）
REST 是一种软件架构风格。这种风格的典型应用，就是 HTTP。其因为简单、扩展性强的特点而广受开发者的青睐。

而 RPC 呢，是 Remote Procedure Call Protocol 的简写，中文描述是远程过程调用，它可以实现客户端像调用本地服务 (方法) 一样调用服务器的服务(方法)。
而 RPC 可以基于 TCP/UDP，也可以基于 HTTP 协议进行传输的，按理说它和 REST 不是一个层面意义上的东西，不应该放在一起讨论，但是谁让 REST 这么流行呢，它是目前最流行的一套互联网应用程序的 API 设计标准，某种意义下，我们说 REST 可以其实就是指代 HTTP 协议。

从使用上来看，HTTP 接口只关注服务提供方，对于客户端怎么调用并不关心。接口只要保证有客户端调用时，返回对应的数据就行了。而 RPC 则要求客户端接口保持和服务端的一致。
REST 是服务端把方法写好，客户端并不知道具体方法。客户端只想获取资源，所以发起 HTTP 请求，而服务端接收到请求后根据 URI 经过一系列的路由才定位到方法上面去 RPC 是服务端提供好方法给客户端调用，客户端需要知道服务端的具体类，具体方法，然后像调用本地方法一样直接调用它。

从设计上来看，RPC，所谓的远程过程调用 ，是面向方法的 ，REST：所谓的 Representational state transfer ，是面向资源的，除此之外，还有一种叫做 SOA，所谓的面向服务的架构，它是面向消息的，这个接触不多，就不多说了。

接口调用通常包含两个部分，序列化和通信协议。
通信协议，上面已经提及了，REST 是 基于 HTTP 协议，而 RPC 可以基于 TCP/UDP，也可以基于 HTTP 协议进行传输的。
常见的序列化协议，有：json、xml、hession、protobuf、thrift、text、bytes 等，REST 通常使用的是 JSON 或者 XML，而 RPC 使用的是 JSON-RPC，或者 XML-RPC。
通过以上几点，我们知道了 REST 和 RPC 之间有很明显的差异。
然后第二个问题：为什么要采用 RPC 呢?
那到底为何要使用 RPC，单纯的依靠 RESTful API 不可以吗？为什么要搞这么多复杂的协议，渣渣表示真的学不过来了。
关于这一点，以下几点仅是我的个人猜想，仅供交流哈：
RPC 和 REST 两者的定位不同，REST 面向资源，更注重接口的规范，因为要保证通用性更强，所以对外最好通过 REST。而 RPC 面向方法，主要用于函数方法的调用，可以适合更复杂通信需求的场景。RESTful API 客户端与服务端之间采用的是同步机制，当发送 HTTP 请求时，客户端需要等待服务端的响应。当然对于这一点是可以通过一些技术来实现异步的机制的。采用 RESTful API，客户端与服务端之间虽然可以独立开发，但还是存在耦合。比如，客户端在发送请求的时，必须知道服务器的地址，且必须保证服务器正常工作。而 rpc + ralbbimq 中间件可以实现低耦合的分布式集群架构。说了这么多，我们该如何选择这两者呢？我总结了如下两点，供你参考：
REST 接口更加规范，通用适配性要求高，建议对外的接口都统一成 REST。而组件内部的各个模块，可以选择 RPC，一个是不用耗费太多精力去开发和维护多套的 HTTP 接口，一个 RPC 的调用性能更高（见下条）从性能角度看，由于 HTTP 本身提供了丰富的状态功能与扩展功能，但也正由于 HTTP 提供的功能过多，导致在网络传输

时，需要携带的信息更多，从性能角度上讲，较为低效。而 RPC 服务网络传输上仅传输与业务内容相关的数据，传输数据更小，性能更高。

为什么一定要 rpc，不能只学 http 协议和 restful 协议吗?

1. rpc 可以基于 tcp 直接开发自己的协议，这个是可以保持长连接的，tcp 的传输效率高，并且可以一直维持链接
2. 自定义协议可以优化数据的传输

如果我们只是开发 web 网站或者一些服务的使用者， 那么我们用 restful 看起来已经足够了，但是 rpc 的这种模式在大量的服务中都有，比如 redis 协议， rabbitmq 的 AMQP 协议， 聊天软件的协议，也就是说我们想要开发一个 redis 的客户端，我们只需要用我们喜欢的语言实现 redis 定义的协议就行了，这对于开发服务来说非常有用，一般这种协议的价值在于我们自己开发的服务之间需要通信的时候 - 那你会问了，自己开发的组件之间协作，直接调用函数不就行了吗？ - 对了，有些人已经反映过来了 – 分布式系统，分布式系统中非常常用， 比如 openstack 中。 还有就是微服务!

所以掌握 rpc 开发，对于进阶和分布式开发就变得非常重要。

http 协议 1.x 一般情况下一个来回就关闭连接，虽然提供了 keep-alive 可以保持长连接，但是依然不方便，所以就出现了 http2.0， http2.0 基本上可以当做 tcp 协议使用了。所以后面讲解到的 grpc 就会使用 http2.0 开发