# 什么是RPC框架?

## RPC（Remote Procedure Call Protocol），网上有各种各样解释RPC的文章，都不够通俗易懂，这是从众多文章中精选出来适合初学者理解的一段话。

早期单机时代，一台电脑上运行多个进程，大家各干各的，老死不相往来。假如A进程需要一个画图的功能，B进程也需要一个画图的功能，程序员就必须为两个进程都写一个画图的功能。这不是整人么？于是就出现了IPC（Inter-process communication，单机中运行的进程之间的相互通信）。OK，现在A既然有了画图的功能，B就调用A进程上的画图功能好了，程序员终于可以偷下懒了。

到了网络时代，大家的电脑都连起来了。以前程序只能调用自己电脑上的进程，能不能调用其他机器上的进程呢？于是就程序员就把IPC扩展到网络上，这就是RPC（远程过程调用）了。现在不仅单机上的进程可以相互通信，多机器中的进程也可以相互通信了。  
  
要知道实现RPC很麻烦呀，什么多线程、什么Socket、什么I/O，都是让咱们普通程序员很头疼的事情。于是就有牛人开发出RPC框架（比如，CORBA、RMI、Web Services、RESTful Web Services等等）。  
  
OK，现在可以定义RPC框架的概念了。简单点讲，RPC框架就是可以让程序员来调用远程进程上的代码一套工具。有了RPC框架，咱程序员就轻松很多了，终于可以逃离多线程、Socket、I/O的苦海了。  
  
关于RPC框架，首先我们要了解什么叫RPC，为什么要用RPC。

RPC是只远程过程调用，也就是说两台服务器A,B, 一个应用部署在A服务器上，另一个应用部署在B服务器上，A服务器上的应用想要调用B服务器上的应用提供的方法/函数，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语意和传递调用的参数。

比如说，一个方法可能这样定义：

Student getStudentByName(String name)

那么：

  首先，要解决通讯的问题，主要是通过在客户端和服务器之间建立TCP连接，远程调用过程中所有交换的数据都在这个连接里传输，连接可以是按需连接，调用结束后就关闭，也

可以是长连接，多个远程调用共享一个连接。

  第二，要解决寻址的问题，也就是说，A服务器上的应用怎么告诉底层的RPC框架，如何连接到B服务器（如主机或IP地址）以及特定的端口号，方法的名称是什么，这样才能完成调用，比如基于WEB服务协议的RPC，就要提供一个endpoint URI，或者是从UDDI服务上查找。如果是RMI调用的话，还需要一个RMI Registry来注册服务的地址。

  第三，当A服务器上的应用发起远程过程调用时，方法的参数需要通过底层的网络协议如TCP传递到B服务器，由于网络协议是基于 二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化（Serialize）或编组(marshal)，通过寻址和传输将序列化的二进制发送给B服务器。

  第四，B服务器收到请求后，需要对参数进行反序列化（序列化的逆操作），恢复为内存中的表达方式，然后找到对应的方法（寻址的一部分）进行本地调用，然后得到返回值。

  第五，返回值还要发送给A服务器上的应用，也要经过序列化的方式发送，服务器A接到后，在反序列化，恢复为内存中的表达方式，交给A服务器上的应用。

