1.REST 和 RESTFUL 是什么

REST（REpresentational State Transfer），State Transfer 为 "状态传输" 或 "状态转移 ",Representational 中文有人翻译为"表征"、"具象", 合起来就是 "表征状态传输" 或 "具象状态传输" 或 "表述性状态转移"

REST 是一种架构风格，REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。其核心是面向资源，REST 专门针对网络应用设计和开发方式，以降低开发的复杂性，提高系统的可伸缩性。

REST 提出设计概念和准则为：

1.网络上的所有事物都可以被抽象为资源(resource)

2.每一个资源都有唯一的资源标识(resource identifier)，对资源的操作不会改变这些标识

3.所有的操作都是无状态的

REST 简化开发，其架构遵循 CRUD 原则，该原则告诉我们对于资源(包括网络资源)只需要四种行为：创建，获取，更新和删除就可以完成相关的操作和处理。您可以通过统一资源标识符（Universal Resource Identifier，URI）来识别和定位资源，并且针对这些资源而执行的操作是通过 HTTP 规范定义的。其核心操作只有 GET,PUT,POST,DELETE。

由于 REST 强制所有的操作都必须是 stateless 的，这就没有上下文的约束，如果做分布式，集群都不需要考虑上下文和会话保持的问题。极大的提高系统的可伸缩性。

2.restful 目前使用现状

目前真正实现 rest 概念的组件比较少，大多数程序员所谓的 rest 只不过是以 GET 访问某个 URL。对 java 而言，后台大多数使用 spring 来处理各类参数及路径参数。但这并不真正符合 Rest 的本义，原始的 Rest 是把所有资源都虚拟化为 URL/URI 来表示，所有操作都使用 HTTP 的几个动作来完成。之前最纯正的 REST 实现是 RestLet，RestLet 基本完全按照作者的博士论文的思路来设计；后边陆续出了一些介于 http 与原始 rest 的实现，前段时间 Linkedin 出了一个 http://Rest.Li，该项目主要是把基于文本的协议修改为二进制协议，效率自然没得说，但引入了过多的复杂性，违背了 rest 简洁的本意。

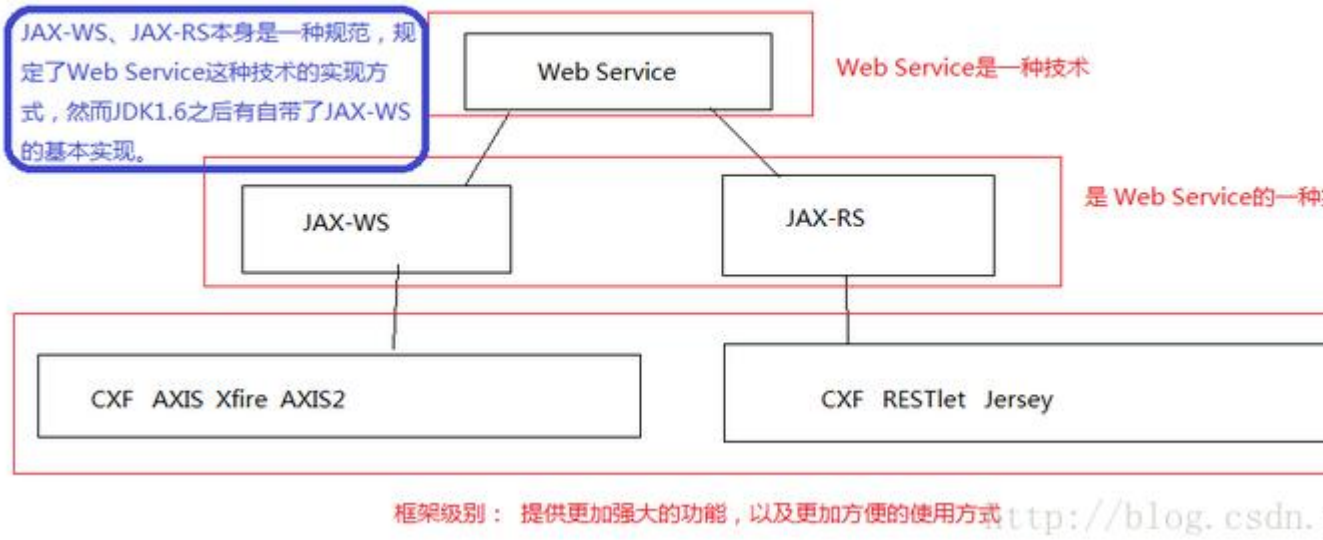
现实中的大多数项目，性能的瓶颈主要集中在后端业务逻辑及算法，协议本身导致性能低下的较少。

3.SOAP Webservice 和 RESTful Webservice

WebService 是一种能够使应用程序在不同的平台使用不同的编程语言进行通讯的技术规范，而这种技术规范的实现可以用不同的方法，比如使用基于 XML 形式的协议（SOAP）进行通讯或者是 RESTFUL 形式的。

既然我们知道可以使用上面的两种形式进行通讯，那么我们就需要对上面的两种形式进行描述，规范化。而这些规范化的工作 sun 已经帮我们完成了，也就是 JAX-WS,JAX-RS 这两种规范。

JAX-WS 是一种规范，而在 jdk1.6 之后就有了自带的实现，但是这种实现是比较简单的，基本上就只能够传递 SOAP 协议格式的消息。这就是为什么我们可以在没有 axis2 或者 CXF 的情况下开发 WebService。 这时候我们就会想了，如果我们需要其他的服务，比如我想让 JAX-WS 与 Spring 集成。这种需求前辈都已经考虑过了，也实现了，不需要我们在去实现这样的需求。而这种需求的解决方案在 JAX-WS 中是采用框架。而 JAX-WS 的框架就有 AXIS2 和 CXF。框架使用起来可能会更加灵活，功能更加强大。比如 CXF 不仅仅实现 JAX-WS，也实现了 JAX-RS 规范。



那么选择 SOAP Webservice 和 Restful Webservice 的使用，首先需要理解就是 SOAP 偏向于面向活动，有严格的规范和标准，包括安全，事务等各个方面的内容，同时 SOAP 强调操作方法和操作对象的分离，有 WSDL 文件规范和 XSD 文件分别对其定义。而 REST 强调面向资源，只要我们要操作的对象可以抽象为资源即可以使用 REST 架构风格。