**socket连接和http连接的区别**

<https://blog.csdn.net/wwd0501/article/details/52412396>

# Socket 与 Http区别 优缺点及应用场景

# <https://blog.csdn.net/ps1397111/article/details/49614319>

Socket适用场景：网络游戏，银行交互，支付。

http适用场景：公司OA服务，互联网服务。

**使用Java内置的Http Server构建Web应用：**

<https://www.cnblogs.com/aspwebchh/p/8300945.html>

创建步骤：

1. 首先要实现 HttpHandler， 重写 handle 方法。（HttpHandler）
2. 创建 web 服务并注册前面创建的 handler，使用 handler 来处理请求。（创建 web 服务）

Eclipse 编译错误 Access restriction:The type \*\*\* is not accessible due to restriction on... 解决方案.

**Eclipse 编译时报错**：   
        Access restriction:The type JPEGCodec is not accessible due to restriction on required library C:/Program Files/Java/jre6/lib/rt.jar   
        **解决方法**：   
        **Project -> Properties -> libraries**，   
        **先 remove 掉 JRE System Library，然后再 Add Library 重新加入**。

支持Http和Https协议,提供了HTTP1.1的部分实现，没有被实现的那部分可以通过扩展已有的Http Server API来实现。程序员必须自己实现HttpHandler接口,HttpServer会调用HttpHandler实现类的回调方法来处理客户端请求,在这里,我们把一个Http请求和它的响应称为一个交换,包装成HttpExchange类,HttpServer负责将HttpExchange传给HttpHandler实现类的回调方法. （摘过来的，也是官方的描述说明）

跨域问题分析：

<https://www.cnblogs.com/lxwphp/p/8080188.html>

java的webservice编写与发布（有问题）:

<https://www.cnblogs.com/xdp-gacl/p/4259481.html>



## [http webservice socket的区别](https://www.cnblogs.com/111testing/p/6581062.html):

<https://www.cnblogs.com/111testing/p/6581062.html>

webservice原理：

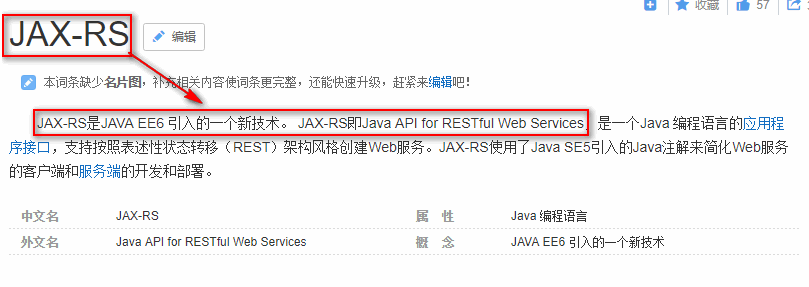
http://www.cnblogs.com/beyondcj/p/6271106.html

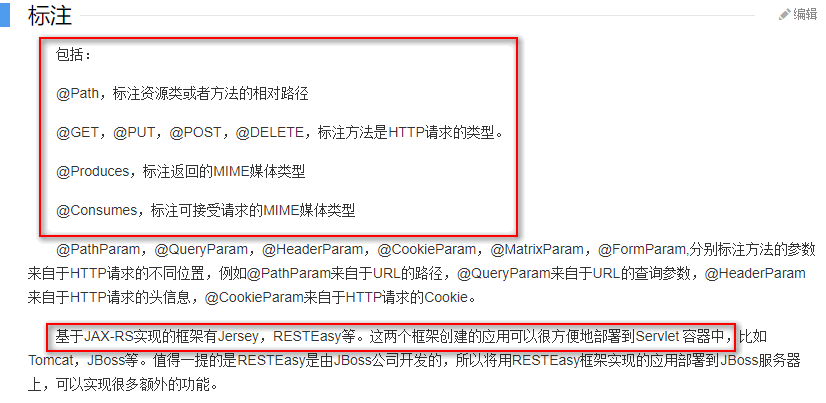
# JAX-WS与JAX-RS区别是什么？

[**https://blog.csdn.net/dogiant/article/details/54907506**](https://blog.csdn.net/dogiant/article/details/54907506)

# JAX-RS（https://www.cnblogs.com/quan-coder/p/8321040.html）

　　　　JAX-RS是JAVA EE6 引入的一个新技术。 JAX-RS即Java API for RESTful Web Services，是一个Java 编程语言的[应用程序接口](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F%E6%8E%A5%E5%8F%A3)，支持按照表述性状态转移（REST）架构风格创建Web服务。JAX-RS使用了Java SE5引入的Java注解来简化Web服务的客户端和[服务端](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E7%AB%AF)的开发和部署。





**注意：**@PathParma、@QueryParam等这些注解，底层都是通过Servlet的方式来获取值的，再结合AOP技术，动态的切入。

|  |  |
| --- | --- |
| 1  2  3  4 | 获取请求方式：  1.  @PathParam    获取路径path上的参数      eg：前台：delete/1001/20   后台：@Path("delete/{id}/{age}") 。获取路径/的值，"/1001/20"既是请求路径的一部分，又是参数值  2.  @QueryParam("id")    获取参数上的参数   eg：前台：delete?id=10， 获取？后面的参数值  3. 请求参数放在body体中   包括form-date、文件、json等。如果用json格式：前台用json格式发送数据，后台用对象来获取，对象的属性与json的key保持一致 |

**webservice**

基于不同的系统之间的数据通信。

jax-ws

基于协议soap ----->xml

jax-rs

基于协议 http ----> json(用的更多) / xml

接口 ，符合restful风格

restful风格特点：

代码层面：一个接口里，无论有多少个方法，只需要提供一个URL，然后通过PUT/GET/POST/DELETE请求格式来确定调用的是哪一个接口方法

使用层面：为了满足给不同的终端提供数据或进行数据通信时，达到一个统一标准，而选择这种接口风格。

--------------------- 本文来自 faultyoung 的CSDN 博客 ，全文地址请点击：<https://blog.csdn.net/faultyoung/article/details/79987307?utm_source=copy>

两类webservice:

[**http://www.360doc.com/content/14/1110/17/203871\_424101460.shtml**](http://www.360doc.com/content/14/1110/17/203871_424101460.shtml)

基于不同的系统之间的数据通信。

jax-ws

基于协议soap ----->xml

jax-rs

基于协议 http ----> json(用的更多) / xml

接口 ，符合restful风格

restful风格特点：

代码层面：一个接口里，无论有多少个方法，只需要提供一个URL，然后通过PUT/GET/POST/DELETE请求格式来确定调用的是哪一个接口方法

使用层面：为了满足给不同的终端提供数据或进行数据通信时，达到一个统一标准，而选择这种接口风格。

--------------------- 本文来自 faultyoung 的CSDN 博客 ，全文地址请点击：<https://blog.csdn.net/faultyoung/article/details/79987307?utm_source=copy>

# [几种流行Webservice框架](https://www.cnblogs.com/beautifulFuture/p/9017967.html)

[**https://www.cnblogs.com/beautifulFuture/p/9017967.html**](https://www.cnblogs.com/beautifulFuture/p/9017967.html)

webservice分类

在SOA的基础技术实现方式中WebService占据了很重要的地位，通常我们提到WebService第一想法就是SOAP消息在各种传输协议上交互。近几年REST的思想伴随着SOA逐渐被大家接受，同时各大网站不断开放API提供给开发者，也激起了REST风格WebService的热潮。

基于soap的webservice

其实SOAP最早是针对RPC的一种解决方案，简单对象访问协议，很轻量，同时作为应用协议可以基于多种传输协议来传递消息（Http,SMTP等）。但是随着SOAP作为WebService的广泛应用，不断地增加附加的内容，使得现在开发人员觉得SOAP很重，使用门槛很高。在SOAP后续的发展过程中，WS-\*一系列协议的制定，增加了SOAP的成熟度，也给SOAP增加了负担。

rest式的webservice

REST其实并不是什么协议也不是什么标准，而是将Http协议的设计初衷作了诠释，在Http协议被广泛利用的今天，越来越多的是将其作为传输协议，而非原先设计者所考虑的应用协议。SOAP类型的WebService就是最好的例子，SOAP消息完全就是将Http协议作为消息承载，以至于对于Http协议中的各种参数（例如编码，错误码等）都置之不顾。其实，最轻量级的应用协议就是Http协议。对于各种资源的操作最后总是能抽象成为这四种基本操作，在定义了定位资源的规则以后，对于资源的操作通过标准的Http协议就可以实现，开发者也会受益于这种轻量级的协议。

restful不一定就要用json，xml同样可以，不过是类似去掉header的soap webservice。要注意的是，就算是restful，大多数公司也没有完全遵守，而且在复杂的业务接口设计下，要通过RUCD来抽象所有的接口作为资源基本是不实际的。很多公司只是套了个rest的壳子，自己在内部实现了别的协议，所以只要能达到目的，无论是基于soap的还是restful的webservice，并不一定都是泾渭分明的。

那该如何选择？

REST对于资源型服务接口来说很合适，同时特别适合对于效率要求很高，但是对于安全要求不高的场景。SOAP的成熟性可以给需要提供给多开发语言的，对于安全性要求较高的接口设计带来便利。

总的来说，基于soap协议的webservice由于有历史包袱，在安全性和扩展性上增强了很多，而且使用xml解析。相应地效率也不如rest式的。但restful的webservice速度虽然可以，但安全性上远不如上一个。所以没必要一定分是非高下，关键还是看应用场景。比如银行国企自然走soap协议的，对安全要求不高的web项目自然去追求速度。

常用的webservice框架

如cfx、jwx、XFire与Axis2，jwx也就是jax-ws，是java6提供的对webservice的一种实现。cxf框架则简化了服务发布过程。Axis与XFire已随着技术不断的更替慢慢落幕，而目前也只有axis2和cxf官方有更新，Axis与XFire都已不再更新。

作者：CodeBuilder  
链接：https://www.jianshu.com/p/4e257042975c  
來源：简书  
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。

https://www.cnblogs.com/zhuyiqizhi/p/6213502.html

## [RESTful Webservice](https://www.cnblogs.com/zhuyiqizhi/p/6213502.html)

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。

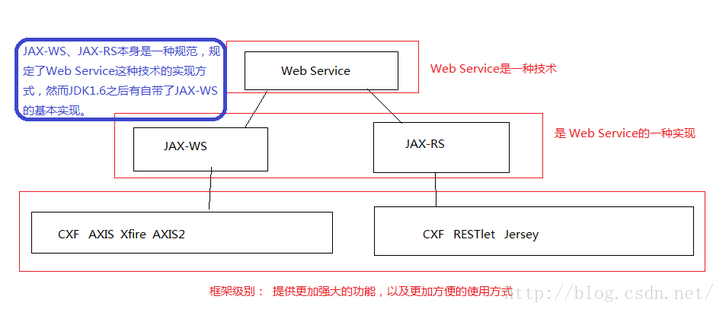
   由于REST强制所有的操作都必须是stateless的，这就没有上下文的约束，如果做分布式，集群都不需要考虑上下文和会话保持的问题。极大的提高系统的可伸缩性。

2.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。

RESTful 风格的 webservice 越来越流行了， sun 也推出了 RESTful WebService 的官方规范： JAX-RS ，全称：

Java API for RESTful WebService。该规范定义了一系列的注解

RESTful 简化了 web service 的设计，它不再需要**wsdl** ，也不再需要 **soap 协议**，而是通过最简单的 http 协议传输数据 ( 包括 xml 或 json) 。既简化了设计，也减少了网络传输量（因为只传输代表数据的 xml 或 json ，没有额外的 xml 包装）。

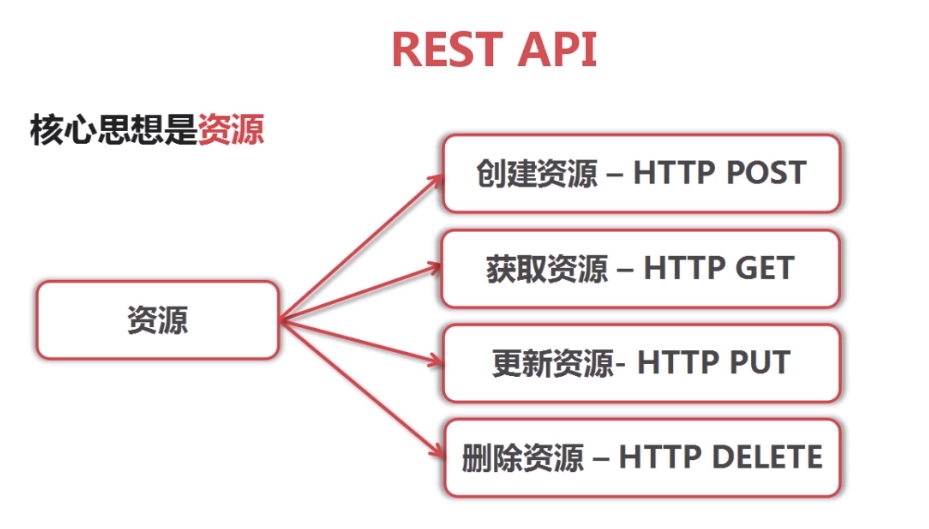


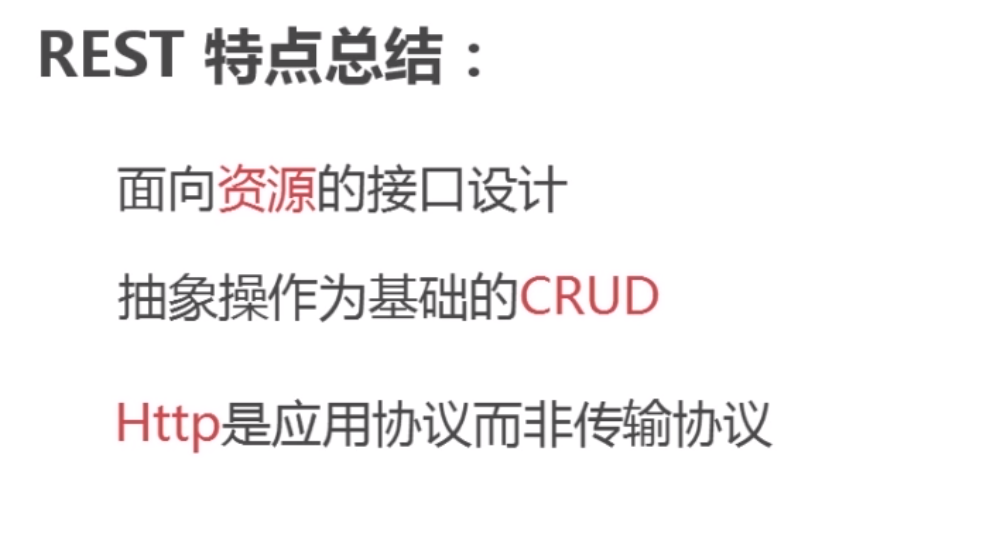


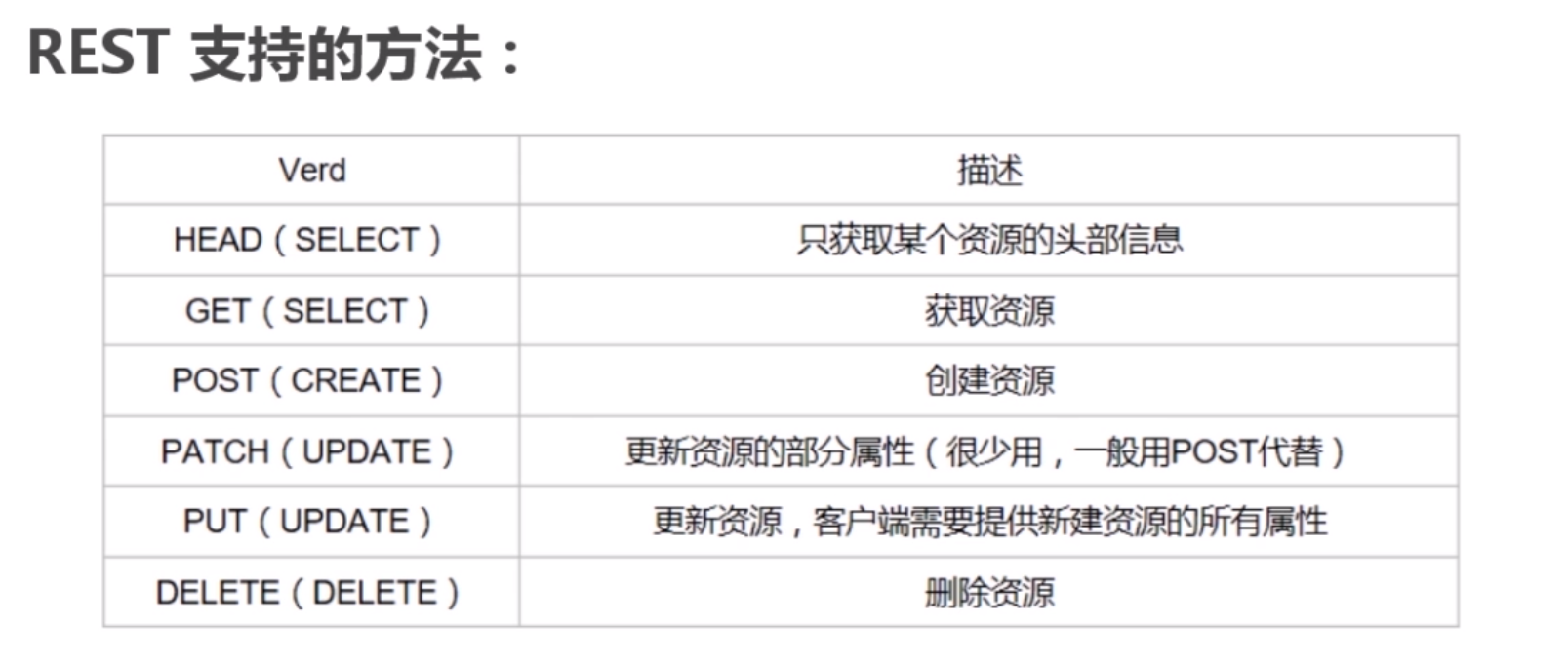














ISO的七层模型 ： 物理层、数据链路层、网络层、传输层、表示层、会话层、应用层

* Socket访问 ： Socket属于传输层，它是对Tcp/ip协议的实现，包含TCP/UDP,它是所有通信协议的基础，Http协议需要Socket支持，以Socket作为基础
* Socket通信特点：
  + 开启端口，该通信是 长连接的通信 ，很容易被防火墙拦截，可以通过心跳机制来实现 ，开发难度大
  + 传输的数据一般是字符串 ，可读性不强
  + socket端口不便于推广
  + 性能相对于其他的通信协议是最优的
* Http协议访问 ：属于应用层的协议，对Socket进行了封装
  + 跨平台
  + 传数据不够友好
  + 对第三方应用提供的服务，希望对外暴露服务接口  
    问题：
* 数据封装不够友好 ：可以用xml封装数据
* 希望给第三方应用提供web方式的服务 （http + xml） = web Service