[Socket 与 WebSocket](https://blog.zengrong.net/post/2199.html)

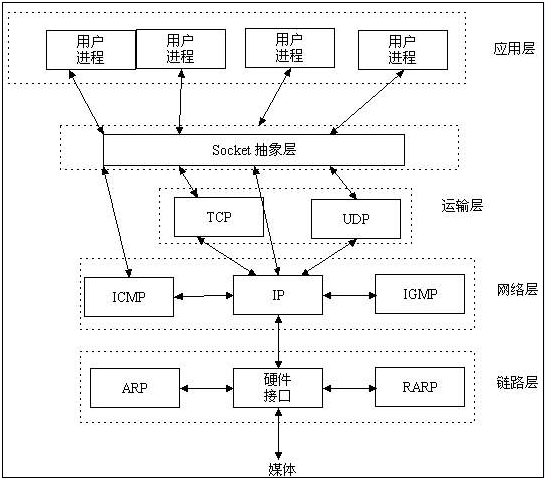
WebSocket协议之前，双工通信是通过多个http链接来实现，这导致了效率低下。WebSocket解决了这个问题。

[WebSocket协议是基于TCP的一种新的网络协议。](https://baike.baidu.com/item/WebSocket/1953845?fr=aladdin)**[它实现了浏览器与服务器全双工(full-duplex)通信](https://baike.baidu.com/item/WebSocket/1953845?fr=aladdin)**[——允许服务器主动发送信息给客户端。](https://baike.baidu.com/item/WebSocket/1953845?fr=aladdin)

[Socket](http://en.wikipedia.org/wiki/Network_socket" \t "https://blog.zengrong.net/post/_blank) 其实并不是一个协议。它工作在 OSI 模型会话层（第5层），是为了方便大家直接使用更底层协议（一般是 [TCP](http://en.wikipedia.org/wiki/Transmission_Control_Protocol" \t "https://blog.zengrong.net/post/_blank) 或 [UDP](http://en.wikipedia.org/wiki/User_Datagram_Protocol" \t "https://blog.zengrong.net/post/_blank) ）而存在的一个抽象层。即[Socket](http://en.wikipedia.org/wiki/Network_socket" \t "https://blog.zengrong.net/post/_blank) 是 [TCP](http://en.wikipedia.org/wiki/Transmission_Control_Protocol" \t "https://blog.zengrong.net/post/_blank) 或 [UDP](http://en.wikipedia.org/wiki/User_Datagram_Protocol" \t "https://blog.zengrong.net/post/_blank)的一种实现。

最早的一套 Socket API 是 [Berkeley sockets](http://en.wikipedia.org/wiki/Berkeley_sockets" \t "https://blog.zengrong.net/post/_blank) ，采用 C 语言实现。它是 Socket 的事实标准，POSIX sockets 是基于它构建的，多种编程语言都遵循这套 API，在 JAVA、Python 中都能看到这套 API 的影子。

**Socket**是应用层与TCP/IP协议族通信的**中间软件抽象层，它是一组接口**。在设计模式中，Socket其实就是一个**门面模式**，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议。



[HTML5 WebSocket](http://www.runoob.com/html/html5-websocket.html)

# [RPC服务和HTTP服务对比](https://blog.csdn.net/wangyunpeng0319/article/details/78651998)

远程过程调用协议RPC（Remote Procedure Call）—[远程过程调用](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8" \t "https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8%E5%8D%8F%E8%AE%AE/_blank)，它是一种通过[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C" \t "https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8%E5%8D%8F%E8%AE%AE/_blank)从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。

### **流行的RPC框架**

目前流行的开源RPC框架还是比较多的。下面重点介绍三种：

1、gRPC是Google最近公布的开源软件，**基于最新的HTTP2.0协议**，并支持常见的众多编程语言。 我们知道HTTP2.0是基于二进制的HTTP协议升级版本，目前各大浏览器都在快马加鞭的加以支持。这个RPC框架是基于HTTP协议实现的，**底层使用到了Netty框架的支持。**

2、Thrift是Facebook的一个开源项目，主要是一个跨语言的服务开发框架。它有一个代码生成器来对它所定义的IDL定义文件自动生成服务代码框架。用户只要在其之前进行二次开发就行，对于底层的RPC通讯等都是透明的。不过这个对于用户来说的话需要学习特定领域语言这个特性，还是有一定成本的。

3、Dubbo是阿里集团开源的一个极为出名的**RPC框架**，在很多互联网公司和企业应用中广泛使用。协议和序列化框架都可以插拔是及其鲜明的特色。同样的远程接口是基于Java Interface，并且依托于spring框架方便开发。可以方便的打包成单一文件，独立进程运行，和现在的微服务概念一致。

偷偷告诉你集团内部已经不怎么使用dubbo啦，现在用的比较多的叫HSF，又名“好舒服”。后面有可能会开源，大家拭目以待。

# **[WEB开发中，使用JSON-RPC好，还是RESTful API好？](https://www.zhihu.com/question/28570307)**

**[Web Service](https://baike.baidu.com/item/Web%20Service/1215039?fr=aladdin&fromid=2342584&fromtitle=webservice)**

## [什么是 Bootstrap？](http://www.runoob.com/bootstrap/bootstrap-intro.html)

Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的。

Tomcat9 新特性：？

# [CA认证](https://baike.baidu.com/item/CA%E8%AE%A4%E8%AF%81/6471579)

**[RSA加密算法](https://baike.baidu.com/item/RSA%E7%AE%97%E6%B3%95/263310?fromtitle=RSA&fromid=210678&fr=aladdin)**是一种[非对称加密算法](https://baike.baidu.com/item/%E9%9D%9E%E5%AF%B9%E7%A7%B0%E5%8A%A0%E5%AF%86%E7%AE%97%E6%B3%95/1208652" \t "https://baike.baidu.com/item/RSA%E7%AE%97%E6%B3%95/_blank)。在[公开密钥加密](https://baike.baidu.com/item/%E5%85%AC%E5%BC%80%E5%AF%86%E9%92%A5%E5%8A%A0%E5%AF%86/8090774" \t "https://baike.baidu.com/item/RSA%E7%AE%97%E6%B3%95/_blank)和[电子商业](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E5%95%86%E4%B8%9A/10778454" \t "https://baike.baidu.com/item/RSA%E7%AE%97%E6%B3%95/_blank)中RSA被广泛使用。

# [axis](https://baike.baidu.com/item/axis/485541?fr=aladdin)

## （阿帕奇可扩展交互系统）

Axis本质上就是一个SOAP引擎，提供创建[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "https://baike.baidu.com/item/axis/_blank)端、客户端和[网关](https://baike.baidu.com/item/%E7%BD%91%E5%85%B3/98992" \t "https://baike.baidu.com/item/axis/_blank)SOAP操作的基本框架。Axis还是一个独立的SOAP服务器和一个嵌入Servlet引擎（例如Tomcat）的服务器。Cxf,xfire.

**myeclipse**->project->**clean**…到底有什么用:

其实主要作用就是**把编译好的class等文件删除，激活eclipse的自动编译**。

解决的问题就是，有时候你把代码改了，但因为一些未知的原因，eclipse的自动编译没有成功，导致运行结果不正常。  
  
当你的工程出现一些莫名其妙的错误时（当然不是语法错误），用一下这个功能，效果不错的

**[HTTP 方法：GET 对比 POST](http://www.w3school.com.cn/tags/html_ref_httpmethods.asp)**

## **比较 GET 与 POST**

下面的表格比较了两种 HTTP 方法：GET 和 POST。

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

**[获取post请求中body的内容](https://www.cnblogs.com/wangguangjian/p/9114659.html)**

# [java通过HttpServletRequest获取post请求中的body内容](https://blog.csdn.net/feinifi/article/details/72990249)

# [使用@RequestBody 取POST方式的json字符串](https://blog.csdn.net/BigBad/article/details/78138547)

HTML 常用标签总结： <http://www.runoob.com/>