



大家都在搜....

Q

下载APP

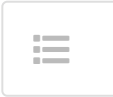
开源软件

问答

动弹

集

- 打赏 ¥
- 评论
- 收藏 ☆
- 点赞
- 分享文章
- 微博
- QQ
- 微信



城市之雾的个人空间 > 后端开发 > 正文

 华为云

B2B·企业上云节

云主机低至**8.28元**/月

立即抢购

B2B
上云节

WebSocket介绍与原理（与http、socket的区别） 转



城市之雾 发布于 2018/04/03 10:15 字数 3498 阅读 275 收藏 0 点赞 0 评论 0

撸了今年阿里、头条和美团的面试，我有一个重要发现.....>>> HOT

WebSocket介绍与原理

WebSocket protocol 是HTML5一种新的协议。它实现了浏览器与服务器全双工通信(full-duplex)。一开始的握手需要借助HTTP请求完成。

——百度百科

目的：即时通讯，替代轮询

网站上的即时通讯是很常见的，比如网页的QQ，聊天系统等。按照以往的技术能力通常是采用轮询、Comet技术解决。

HTTP协议是非持久化的，单向的网络协议，在建立连接后只允许浏览器向服务器发出请求后，服务器才能返回相应的数据。当需要即时通讯时，通过轮询在特定的时间间隔（如1秒），由浏览器向服务器发送Request请求，然后将最新的数据返回给浏览器。这样的方法最明显的缺点就是需要不断的发送请求，而且通常HTTP request的Header是非常长的，为了传输一个很小的数据 需要付出巨大的代价，是很不合算的，占用了很多的宽带。

缺点：会导致过多不必要的请求，浪费流量和服务器资源，每一次请求、应答，都浪费了一定流量在相同的头部信息上

然而WebSocket的出现可以弥补这一缺点。在WebSocket中，只需要服务器和浏览器通过HTTP协议进行一个握手的动作，然后单独建立一条TCP的通信通道进行数据的传送。

原理

WebSocket同HTTP一样也是应用层的协议，但是它是一种双向通信协议，是建立在TCP之上的。

连接过程 —— 握手过程

- 1. 浏览器、服务器建立TCP连接，三次握手。这是通信的基础，传输控制层，若失败后续都不执行。
- 2. TCP连接成功后，浏览器通过HTTP协议向服务器传送WebSocket支持的版本号等信息。（开始前的**HTTP**握手）
- 3. 服务器收到客户端的握手请求后，同样采用**HTTP**协议回馈数据。
- 4. 当收到了连接成功的消息后，通过TCP通道进行传输通信。

WebSocket与HTTP的关系

相同点

- 1. 都是一样基于TCP的，都是可靠性传输协议。
- 2. 都是应用层协议。

不同点

- 1. WebSocket是双向通信协议，模拟Socket协议，可以双向发送或接受信息。HTTP是单向的。
- 2. WebSocket是需要握手进行建立连接的。

联系

WebSocket在建立握手时，数据是通过HTTP传输的。但是建立之后，在真正传输时候是不需要HTTP协议的。

WebSocket与Socket的关系

Socket其实并不是一个协议，而是为了方便使用TCP或UDP而抽象出来的一层，是位于应用层和传输控制层之间的一组接口。

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。

当两台主机通信时，必须通过Socket连接，Socket则利用TCP/IP协议建立TCP连接。TCP连接则更依赖于底层的IP协议，IP协议的连接则依赖于链路层等更低层次。

WebSocket则是一个典型的应用层协议。

区别

Socket是传输控制层协议，**WebSocket**是应用层协议。

HTML5与WebSocket的关系

WebSocket API 是 HTML5 标准的一部分，但这并不代表 WebSocket 一定要用在 HTML 中，或者只能在基于浏览器的应用程序中使用。

实际上，许多语言、框架和服务器都提供了 WebSocket 支持，例如：

- * 基于 C 的 libwebsocket.org

- * 基于 Node.js 的 Socket.io
- * 基于 Python 的 ws4py
- * 基于 C++ 的 WebSocket++
- * Apache 对 WebSocket 的支持：Apache Module mod_proxy_wstunnel
- * Nginx 对 WebSockets 的支持：NGINX as a WebSockets Proxy 、 NGINX Announces Support for WebSocket Protocol 、 WebSocket proxying
- * lighttpd 对 WebSocket 的支持：mod_websocket

WebSocket 机制

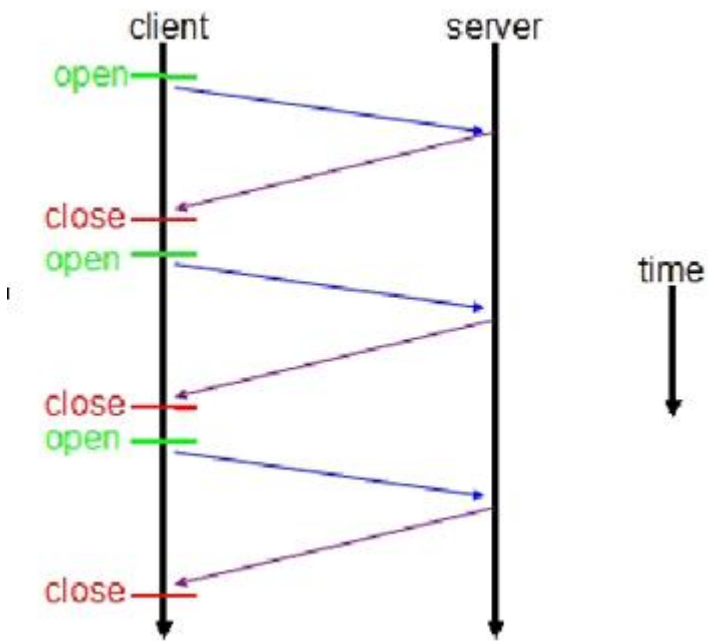
以下简要介绍一下 WebSocket 的原理及运行机制。

WebSocket 是 HTML5 一种新的协议。它实现了浏览器与服务器全双工通信，能更好的节省服务器资源和带宽并达到实时通讯，它建立在 TCP 之上，同 HTTP 一样通过 TCP 来传输数据，但是它和 HTTP 最大不同是：

- WebSocket 是一种双向通信协议，在建立连接后，WebSocket 服务器和 Browser/Client Agent 都能主动的向对方发送或接收数据，就像 Socket 一样；
- WebSocket 需要类似 TCP 的客户端和服务端通过握手连接，连接成功后才能相互通信。

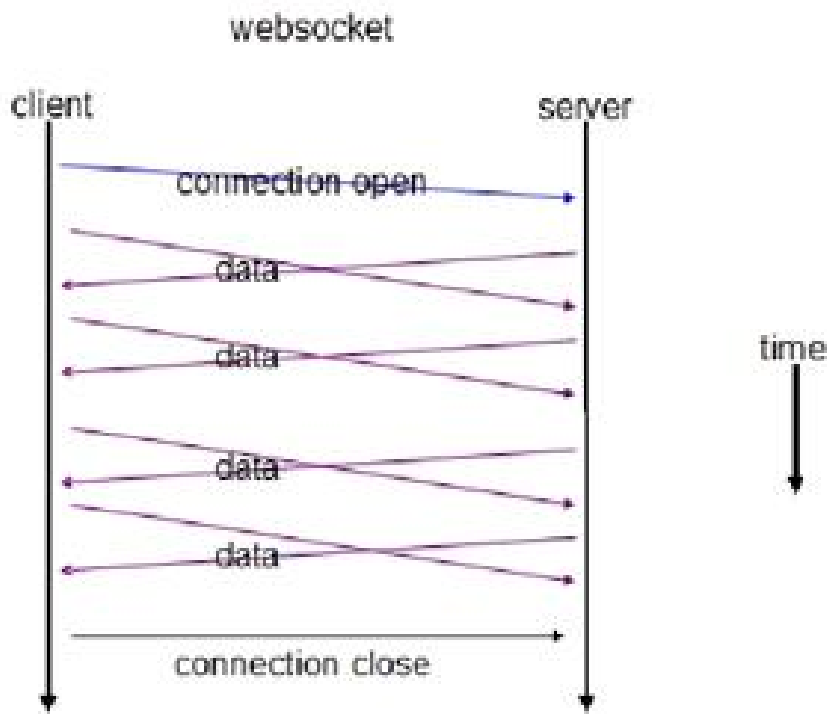
非 WebSocket 模式传统 HTTP 客户端与服务器的交互如下图所示：

图 1. 传统 HTTP 请求响应客户端服务器交互图



使用 WebSocket 模式客户端与服务器的交互如下图：

图 2.WebSocket 请求响应客户端服务器交互图



上图对比可以看出，相对于传统 HTTP 每次请求-应答都需要客户端与服务端建立连接的模式，WebSocket 是类似 Socket 的 TCP 长连接的通讯模式，一旦 WebSocket 连接建立后，后续数据都以帧序列的形式传输。在客户端断开 WebSocket 连接或 Server 端断掉连接前，不需要客户端和服务端重新发起连接请求。在海量并发及客户端与服务器交互负载流量大的情况下，极大的节省了网络带宽资源的消耗，有明显的性能优势，且客户端发送和接受消息是在同一个持久连接上发起，实时性优势明显。

我们再通过客户端和服务端交互的报文看一下 WebSocket 通讯与传统 HTTP 的不同：

在客户端，new WebSocket 实例化一个新的 WebSocket 客户端对象，连接类似 ws://yourdomain:port/path 的服务端 WebSocket URL，WebSocket 客户端对象会自动解析并识别为 WebSocket 请求，从而连接服务端端口，执行双方握手过程，客户端发送数据格式类似：

清单 1.WebSocket 客户端连接报文

```
GET /webfin/websocket/ HTTP/1.1
Host: localhost
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: xqBt3ImNzJbYqRINxEF1kg==
Origin:
http://localhost
:8080
Sec-WebSocket-Version: 13
```

可以看到，客户端发起的 WebSocket 连接报文类似传统 HTTP 报文，“Upgrade: websocket”参数值表明这是 WebSocket 类型请求，“Sec-WebSocket-Key”是 WebSocket 客户端发送的一个 base64 编码的密文，要求服务端必须返回一个对应加密的“Sec-WebSocket-Accept”应答，否则客户端会抛出“Error during WebSocket handshake”错误，并关闭连接。

服务端收到报文后返回的数据格式类似：

清单 2.WebSocket 服务端响应报文

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: K7DJLdLooIwIG/M0pvWFB3y3FE8=
```

“Sec-WebSocket-Accept”的值是服务端采用与客户端一致的密钥计算出来后返回客户端的，“HTTP/1.1 101 Switching Protocols”表示服务端接受 WebSocket 协议的客户端连接，经过这样的请求-响应处理后，客户端服务端的 WebSocket 连接握手成功，后续就可以进行 TCP 通讯了。

在开发方面，WebSocket API 也十分简单，我们只需要实例化 WebSocket，创建连接，然后服务端和客户端就可以相互发送和响应消息，在下文 WebSocket 实现及案例分析部分，可以看到详细的 WebSocket API 及代码实现。

WebSocket 实现

如上文所述，WebSocket 的实现分为客户端和服务端两部分，客户端（通常为浏览器）发出 WebSocket 连接请求，服务端响应，实现类似 TCP 握手的动作，从而在浏览器客户端和 WebSocket 服务端之间形成一条 HTTP 长连接快速通道。两者之间后续进行直接的数据互相传送，不再需要发起连接和相应。

以下简要描述 WebSocket 服务端 API 及客户端 API。

WebSocket 服务端 API

WebSocket 服务端在各个主流应用服务器厂商中已基本获得符合 JEE JSR356 标准规范 API 的支持，以下列举了部分常见的商用及开源应用服务器对 WebSocket Server 端的支持情况：

表 1.WebSocket 服务端支持

厂商	应用服务器	备注
IBM	WebSphere	WebSphere 8.0 以上版本支持，7.X 之前版本结合 MQTT 支持类似的 HTTP 长连接
甲骨文	WebLogic	WebLogic 12c 支持，11g 及 10g 版本通过 HTTP Publish 支持类似的 HTTP 长连接
微软	IIS	IIS 7.0+支持
Apache	Tomcat	Tomcat 7.0.5+支持，7.0.2X 及 7.0.3X 通过自定义 API 支持
	Jetty	Jetty 7.0+支持

以下我们使用 Tomcat7.0.5 版本的服务端示例代码说明 WebSocket 服务端的实现：

JSR356 的 WebSocket 规范使用 javax.websocket.*的 API，可以将一个普通 Java 对象（POJO）使用 @ServerEndpoint 注释作为 WebSocket 服务器的端点，代码示例如下：

清单 3.WebSocket 服务端 API 示例

```
@ServerEndpoint("/echo")
public class EchoEndpoint {

    @OnOpen
    public void onOpen(Session session) throws IOException {
        //以下代码省略...
    }

    @OnMessage
    public String onMessage(String message) {
        //以下代码省略...
    }
}
```



```
@Message(maxMessageSize=6)
public void receiveMessage(String s) {
    //以下代码省略...
}

@OnError
public void onError(Throwable t) {
    //以下代码省略...
}

@OnClose
public void onClose(Session session, CloseReason reason) {
    //以下代码省略...
}

}
```

代码解释：

上文的简洁代码即建立了一个 WebSocket 的服务端，@ServerEndpoint("/echo") 的 annotation 注释端点表示将 WebSocket 服务端运行在 ws://[Server 端 IP 或域名]:[Server 端口]/websockets/echo 的访问端点，客户端浏览器已经可以对 WebSocket 客户端 API 发起 HTTP 长连接了。

使用 ServerEndpoint 注释的类必须有一个公共的无参数构造函数，@onMessage 注解的 Java 方法用于接收传入的 WebSocket 信息，这个信息可以是文本格式，也可以是二进制格式。

OnOpen 在这个端点一个新的连接建立时被调用。参数提供了连接的另一端的更多细节。Session 表明两个 WebSocket 端点对话连接的另一端，可以理解为类似 HTTPSession 的概念。

OnClose 在连接被终止时调用。参数 closeReason 可封装更多细节，如为什么一个 WebSocket 连接关闭。

更高级的定制如 @Message 注释，MaxMessageSize 属性可以被用来定义消息字节最大限制，在示例程序中，如果超过 6 个字节的信息被接收，就报告错误和连接关闭。

注意：早期不同应用服务器支持的 WebSocket 方式不尽相同，即使同一厂商，不同版本也有细微差别，如 Tomcat 服务器 7.0.5 以上的版本都是标准 JSR356 规范实现，而 7.0.2x/7.0.3X 的版本使用自定义 API（WebSocketServlet 和 StreamInbound，前者是一个容器，用来初始化 WebSocket 环境；后者是用来具体处理 WebSocket 请求和响应，详见案例分析部分），且 Tomcat7.0.3x 与 7.0.2x 的 createWebSocketInbound 方法的定义不同，增加了一个 HttpServletRequest 参数，使得可以从 request 参数中获取更多 WebSocket 客户端的信息，如下代码所示：

清单 4.Tomcat7.0.3X 版本 WebSocket API

```
public class EchoServlet extends WebSocketServlet {
    @Override
    protected StreamInbound createWebSocketInbound(String subProtocol,
        HttpServletRequest request) {
        //以下代码省略....
        return new MessageInbound() {
            //以下代码省略....
        }
    }
    protected void onBinaryMessage(ByteBuffer buffer)
        throws IOException {
        //以下代码省略...
    }
    protected void onTextMessage(CharBuffer buffer) throws IOException {
        getWsOutbound().writeTextMessage(buffer);
        //以下代码省略...
    }
}
};
}
```

因此选择 WebSocket 的 Server 端重点需要选择其版本，通常情况下，更新的版本对 WebSocket 的支持是标准 JSR 规范 API，但也要考虑开发易用性及老版本程序移植性等方面的问题，如下文所述的客户案例，就是因为客户要求统一应用服务器版本所以使用的 Tomcat 7.0.3X 版本的 WebSocketServlet 实现，而不是 JSR356 的 @ServerEndpoint 注释端点。

WebSocket 客户端 API

对于 WebSocket 客户端，主流的浏览器（包括 PC 和移动终端）现已都支持标准的 HTML5 的 WebSocket API，这意味着客户端的 WebSocket JavaScript 脚本具备良好的一致性和跨平台特性，以下列举了常见的浏览器厂商对 WebSocket 的支持情况：

表 2.WebSocket 客户端支持

浏览器	支持情况
Chrome	Chrome version 4+支持
Firefox	Firefox version 5+支持
IE	IE version 10+支持
Safari	IOS 5+支持
Android Brower	Android 4.5+支持

客户端 WebSocket API 基本上已经在各个主流浏览器厂商中实现了统一，因此使用标准 HTML5 定义的 WebSocket 客户端的 JavaScript API 即可，当然也可以使用业界满足 WebSocket 标准规范的开源

框架，如 Socket.io。

以下以一段代码示例说明 WebSocket 的客户端实现：

清单 5.WebSocket 客户端 API 示例

```
var ws = new WebSocket("ws://echo.websocket.org");
ws.onopen = function(){ws.send("Test!"); };
ws.onmessage = function(evt){console.log(evt.data);ws.close();};
ws.onclose = function(evt){console.log("WebSocketClosed!");};
ws.onerror = function(evt){console.log("WebSocketError!");};
```

第一行代码是在申请一个 WebSocket 对象，参数是需要连接的服务器端的地址，同 HTTP 协议开头一样，WebSocket 协议的 URL 使用 ws://开头，另外安全的 WebSocket 协议使用 wss://开头。

第二行到第五行为 WebSocket 对象注册消息的处理函数，WebSocket 对象一共支持四个消息 onopen, onmessage, onclose 和 onerror，有了这 4 个事件，我们就可以很容易很轻松的驾驭 WebSocket。

当 Browser 和 WebSocketServer 连接成功后，会触发 onopen 消息；如果连接失败，发送、接收数据失败或者处理数据出现错误，browser 会触发 onerror 消息；当 Browser 接收到 WebSocketServer 发送过来的数据时，就会触发 onmessage 消息，参数 evt 中包含 Server 传输过来的数据；当 Browser 接收到 WebSocketServer 端发送的关闭连接请求时，就会触发 onclose 消息。我们可以看出所有的操作都是采用异步回调的方式触发，这样不会阻塞 UI，可以获得更快的响应时间，更好的用户体验。

Elastic Search Solutions

Powerful search-as-a-service solution for building world-class search experiences. [elastic.co](https://www.elastic.co)

SIGN UP

广告

本文转载自：<https://blog.csdn.net/wwd0501/article/details/54582912>

¥ 打赏

👍 点赞 (0)

☆ 收藏 (0)

➦ 分享

🖨 打印 🚩 举报

◀ 上一篇：[phpStudy项目目录无法访问（报错500...](#)

下一篇：[Beyond Compare常用设置](#) ▶

城市之雾

2019/8/29

WebSocket介绍与原理（与http、socket的区别） - 城市之雾的个人空间 - OSCHINA



粉丝 3 博文 164 码字总数 21199 作品 0

无锡

关注

私信

提问

社区活跃度

学习积极性

社区影响力

开源贡献度

技术贡献度

活动活跃性

相关文章

最新文章

八问WebSocket协议：为你快速解答WebSocket热门疑问

本文由“小姐姐养的狗”原创发布于“小姐姐味道”公众号，原题《WebSocket协议 8 问》，收录时有优化和改动。感谢原作者的分享。一、引言 WebSocket是一种比较新的协议，它是伴随…

备注
连接建立时触发
客户端接收服务端数据时触
通信发生错误时触发
连接关闭时触发

JackJiang2011 04/25 0 0

八问WebSocket协议：为你快速解答WebSocket热门疑问

一、引言 WebSocket是一种比较新的协议，它是伴随着html5规范而生的，虽然还比较年轻，但大多主流浏览器都已经支持。它使用方面、应用广泛，已经渗透到前后端开发的各种场景中。对http一问一…

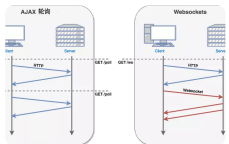
首席大胸器 04/25 90 0

用jetty搭建websocket服务并与ie78兼容的方法

jetty8中已经自带有websocket功能，所以我们可以很方便搭建一个自己的websocket服务。源程序：
http://sdrv.ms/N5BuKw 启动类：org.noahx.websocket.WebSocketServer 访问地址：http://127....

NoahX 2012/08/09 4.2K 6

基于Socket通讯(C#)和WebSocket协议(net)编写的两种聊天功能(文末附源码下载地址)



今天我们来盘一盘Socket通讯和WebSocket协议在即时通讯的小应用——聊天。理论大家估计都知道得差不多了，小编也通过查阅各种资料对理论知识进行了充电，发现好多demo似懂非懂，…


学习中的苦与乐 01/30 0 0

websocket与node.js的完美结合

本文为原创文章，出自http://cnodejs.org，转载请注明出处和作者 作者：kongwu 原文：
http://cnodejs.org/blog/?p=273 之所以写下此文，是我觉得越是简单的技术往往能发挥越重要的作用，随着…

红薯 2011/03/20 10.1K 2

加载更多



「百

智能云

百度智

高稳定云
主机,3
带宽,建
小时
收录,5
稳固,定
份数据
备份,i
线路免

打印

OSCHINA 社区

关于我们
联系我们
合作伙伴
Open API

在线工具

码云 Gitee.com
企业研发管理
CopyCat-代码克隆检测
实用在线工具

微信公众号



OSCHINA APP

聚合全网技术文章，根据你的阅读喜好进行个性推荐

下载 APP