# 【第12章-WebSocket】

如果你有跟进Web 技术的最新进展，你很可能就遇到过“实时Web”这个短语，而如果你在工程领域中有实时应用程序的实战经验，那么你可能有点怀疑这个术语到底意味着什么。

因此，让我们首先澄清，这里并不是指所谓的硬实时服务质量（QoS），硬实时服务质量是保证计算结果将在指定的时间间隔内被递交。仅HTTP 的请求/响应模式设计就使得其很难被支持，从过去所设计的各种方案中都没有提供一种能够提供令人满意的解决方案的事实中便可见一斑。

虽然已经有了一些关于正式定义实时Web服务语义的学术讨论，但是被普遍接受的定义似乎还未出现。因此现在我们将采纳下面来自维基百科的非权威性描述：

**实时Web 利用技术和实践，使用户在信息的作者发布信息之后就能够立即收到信息，而不需要他们或者他们的软件周期性地检查信息源以获取更新。**

简而言之，虽然全面的实时Web可能并不会马上到来，但是它背后的想法却助长了对于几乎瞬时获得信息的期望。我们将在本章中讨论的WebSocket协议便是在这个方向上迈出的坚实的一步。

## 12.1 WebSocket 简介

WebSocket 协议是完全重新设计的协议，旨在为Web 上的双向数据传输问题提供一个切实可行的解决方案，使得客户端和服务器之间可以在任意时刻传输消息，因此，这也就要求它们异步地处理消息回执。（作为HTML5 客户端API 的一部分，大部分最新的浏览器都已经支持了WebSocket。）

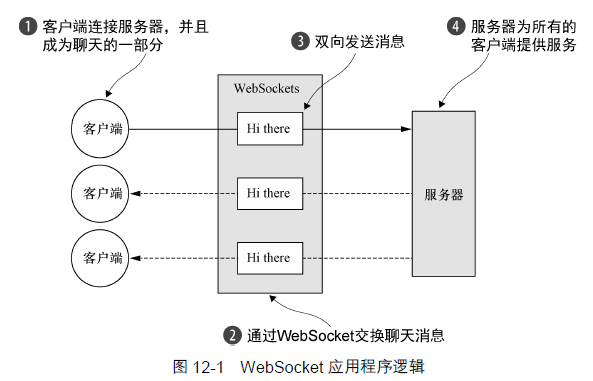
Netty 对于WebSocket 的支持包含了所有正在使用中的主要实现，因此在你的下一个应用程序中采用它将是简单直接的。和往常使用Netty 一样，你可以完全使用该协议，而无需关心它内部的实现细节。我们将通过创建一个基于WebSocket 的实时聊天应用程序来演示这一点。

## 12.2 我们的WebSocket 示例应用程序

为了让示例应用程序展示它的实时功能，我们将通过使用WebSocket 协议来实现一个基于浏览器的聊天应用程序，就像你可能在Facebook 的文本消息功能中见到过的那样。我们将通过使得多个用户之间可以同时进行相互通信，从而更进一步。

图12-1 说明了该应用程序的逻辑：

1. 客户端发送一个消息；
2. 该消息将被广播到所有其他连接的客户端。

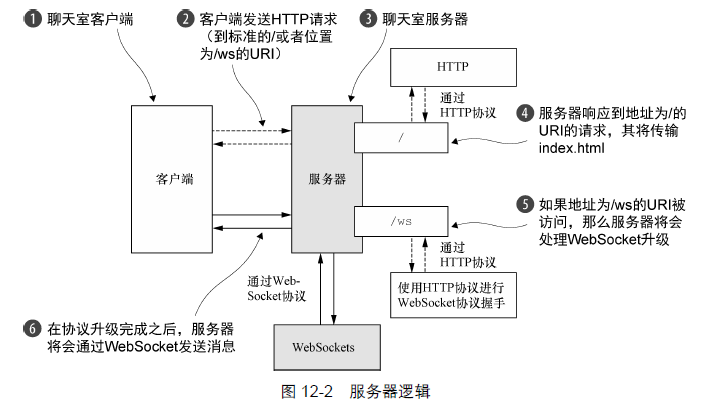


这正如你可能会预期的一个聊天室应当的工作方式：所有的人都可以和其他的人聊天。在示例中，我们将只实现服务器端，而客户端则是通过Web 页面访问该聊天室的浏览器。正如同你将在接下来的几页中所看到的，WebSocket 简化了编写这样的服务器的过程。

## 12.3 添加WebSocket 支持

在从标准的HTTP或者HTTPS协议切换到WebSocket时，将会使用一种称为升级握手的机制。因此，使用WebSocket的应用程序将始终以HTTP/S作为开始，然后再执行升级。这个升级动作发生的确切时刻特定于应用程序；它可能会发生在启动时，也可能会发生在请求了某个特定的URL之后。

我们的应用程序将采用下面的约定：如果被请求的URL 以/ws 结尾，那么我们将会把该协议升级为WebSocket；否则，服务器将使用基本的HTTP/S。在连接已经升级完成之后，所有数据都将会使用WebSocket 进行传输。图12-2 说明了该服务器逻辑，一如在Netty 中一样，它由一组ChannelHandler 实现。我们将会在下一节中，解释用于处理HTTP 以及WebSocket 协议的技术时，描述它们。



### 12.3.1 处理HTTP 请求

首先，我们将实现该处理HTTP 请求的组件。这个组件将提供用于访问聊天室并显示由连接的客户端发送的消息的网页。代码清单12-1 给出了这个HttpRequestHandler 对应的代码，其扩展了SimpleChannelInboundHandler 以处理FullHttpRequest 消息。需要注意的是，channelRead0()方法的实现是如何转发任何目标URI 为/ws 的请求的。