# Boost—Asio

使用asio 需要引用一个头文件#include<boost/asio>

对于所有使用 Asio 的程序，都必须要包含至少一个 io\_service 对象。对于 Asio 这个 Boost 库而言，它抽象了诸如网络、串口通信等等这些概念，并将其统一规为 IO 操作，所以 io\_service 这个类提供了访问 I/O 的功能。因此，使用 Asio 时，必须定义：

boost::asio::io\_service io;

## HTTP

我们知道，HTTP 和 HTTPS 的底层实际上是使用的 TCP 可靠连接，通过 Socket 技术进行通信，而一个 Socket 由 IP 地址及端口构成。无例外地，Asio 同样也需要建立一个和 socket 有关的对象，那就是 boost::asio::ip::tcp::socket。可想而知，Socket 既然是网络通信的基础，那么自然的我们要进行的 IO 操作也就必须在这里完成，因此，我们定义的 boost::asio::ip::tcp::socket 对象，必须由 io\_service 来进行构造，即：

**boost::asio::ip::tcp::socket socket(io);**

不难看出，一个普通的 boost::asio::ip::tcp::socket 对象，实际上就是一个 HTTP 的 Socket 连接，因此我们在日后进行代码编写时，甚至于可以使用 typedef 将这个类型直接定义为 HTTP:

**typedef boost::asio::ip::tcp::socket HTTP;**

然而，作为服务端，我们可能构建很多很多的连接从而响应并发，所以当我们需要建立连接时候，就需要使用一个叫做 acceptor 的对象。

而 boost::asio::ip::tcp::acceptor 从名字上就可以看出，这个对象应该被用于建立连接。在 Boost 中，我们需要初始化一个 acceptor 对象时，必须提供一个 io\_service 对象和一个 endpoint 对象。

那么 endpoint 又是什么？事实上，socket 是一个端到端的连接，所谓 endpoint 就是 socket 位于服务端的一个端点，我们知道，socket 是由 IP 地址和端口号组成的，那么当我们需要为其建立一个 IPv4 的网络，首先可以建立一个 boost::asio::ip::tcp::endpoint 对象：

**unsigned short port = 8080;**

**boost::asio::ip::tcp::endpoint endpoint(boost::asio::ip::tcp::v4(), port);**

其中 boost::asio::ip::tcp::v4() 用于初始化一个 IPv4 的网络。最后在使用这个 endpoint 对象来初始化 acceptor:

**boost::asio::ip::tcp::acceptor acceptor(io, endpoint);**