Qt网络编程

Qt网络模块提供允许您编写TCP / IP客户端和服务器的类。它提供了较低级别的类，如QTcpSocket，QTcpServer和QUdpSocket，它们代表低级网络概念，以及高级类，如QNetworkRequest，QNetworkReply和QNetworkAccessManager，使用通用协议执行网络操作。它还提供实现承载管理的QNetworkConfiguration，QNetworkConfigurationManager和QNetworkSession等类。

Qt的网络编程类

Qt Network C ++ Classes页面包含Qt Network中的C ++类列表。

HTTP和FTP的高级网络操作

Network Access API是用于执行常见网络操作的类的集合。 API为所使用的特定操作和协议提供了一个抽象层（例如，通过HTTP获取和发布数据），并且仅为一般或高级概念公开类，函数和信号。

网络请求由QNetworkRequest类表示，该类还充当与请求相关联的信息的通用容器，例如任何标头信息和使用的加密。构造请求对象时指定的URL确定用于请求的协议。目前支持上传和下载HTTP，FTP和本地文件URL。

网络操作的协调由QNetworkAccessManager类执行。创建请求后，此类用于调度它并发出信号以报告其进度。管理器还协调使用cookie来存储客户端上的数据，身份验证请求以及代理的使用。

对网络请求的回复由QNetworkReply类表示;这些是在调度请求时由QNetworkAccessManager创建的。 QNetworkReply提供的信号可用于单独监控每个回复，或者开发人员可以选择使用管理器的信号来代替，并放弃对回复的引用。由于QNetworkReply是QIODevice的子类，因此可以同步或异步处理回复;即，作为阻塞或非阻塞操作。

每个应用程序或库都可以创建一个或多个QNetworkAccessManager实例来处理网络通信。

将TCP与QTcpSocket和QTcpServer一起使用

TCP（传输控制协议）是大多数Internet协议（包括HTTP和FTP）用于数据传输的低级网络协议。它是一种可靠的，面向流的，面向连接的传输协议。它特别适合连续传输数据。

￼

QTcpSocket类为TCP提供接口。您可以使用QTcpSocket实现标准网络协议，如POP3，SMTP和NNTP，以及自定义协议。

在开始任何数据传输之前，必须建立到远程主机和端口的TCP连接。建立连接后，可通过QTcpSocket :: peerAddress（）和QTcpSocket :: peerPort（）获得对等体的IP地址和端口。任何时候，对等方都可以关闭连接，然后数据传输将立即停止。

QTcpSocket异步工作并发出信号以报告状态更改和错误，就像QNetworkAccessManager一样。它依赖于事件循环来检测传入数据并自动刷新传出数据。您可以使用QTcpSocket :: write（）将数据写入套接字，并使用QTcpSocket :: read（）读取数据。 QTcpSocket代表两个独立的数据流：一个用于读取，一个用于写入。

由于QTcpSocket继承了QIODevice，因此可以将它与QTextStream和QDataStream一起使用。从QTcpSocket读取时，必须事先通过调用QTcpSocket :: bytesAvailable（）来确保有足够的数据可用。

如果需要处理传入的TCP连接（例如，在服务器应用程序中），请使用QTcpServer类。调用QTcpServer :: listen（）来设置服务器，并连接到QTcpServer :: newConnection（）信号，该信号为每个连接的客户端发出一次。在您的插槽中，调用QTcpServer :: nextPendingConnection（）以接受连接并使用返回的QTcpSocket与客户端进行通信。

虽然它的大部分功能都是异步工作的，但可以同步使用QTcpSocket（即阻塞）。要获得阻塞行为，请调用QTcpSocket的waitFor ...（）函数;这些挂起调用线程，直到发出信号。例如，在调用非阻塞QTcpSocket :: connectToHost（）函数之后，调用QTcpSocket :: waitForConnected（）来阻塞线程，直到发出connected（）信号。

同步套接字通常会导致代码具有更简单的控制流。 waitFor ...（）方法的主要缺点是在waitFor ...（）函数阻塞时不会处理事件。如果在GUI线程中使用，则可能会冻结应用程序的用户界面。因此，我们建议您仅在非GUI线程中使用同步套接字。同步使用时，QTcpSocket不需要事件循环。

Fortune Client和Fortune Server示例显示了如何使用QTcpSocket和QTcpServer编写TCP客户端 - 服务器应用程序。另请参阅阻止Fortune客户端以获取有关如何在单独线程中使用同步QTcpSocket的示例（不使用事件循环），以及用于多线程TCP服务器的示例的Threaded Fortune Server，每个活动客户端具有一个线程。

使用UDP和QUdpSocket

UDP（用户数据报协议）是一种轻量级，不可靠，面向数据报的无连接协议。当可靠性不重要时，可以使用它。例如，报告时间的服务器可以选择UDP。如果丢失了具有时间的数据报，则客户端可以简单地发出另一个请求。

QUdpSocket类允许您发送和接收UDP数据报。它继承了QAbstractSocket，因此它共享QTcpSocket的大部分接口。主要区别在于QUdpSocket将数据传输为数据报而不是连续的数据流。简而言之，数据报是有限大小的数据包（通常小于512字节），除了传输的数据外，还包含数据报的发送方和接收方的IP地址和端口。

QUdpSocket支持IPv4广播。广播通常用于实现网络发现协议，例如查找网络上哪个主机具有最多的可用硬盘空间。一台主机向所有其他主机接收的网络广播数据报。接收请求的每个主机然后使用其当前可用磁盘空间量将回复发送回发送方。发起方等待它收到来自所有主机的回复，然后可以选择具有最多可用空间的服务器来存储数据。要广播数据报，只需将其发送到特殊地址QHostAddress :: Broadcast（255.255.255.255），或发送到本地网络的广播地址。

QUdpSocket :: bind（）准备套接字以接受传入的数据报，就像TCP服务器的QTcpServer :: listen（）一样。每当一个或多个数据报到达时，QUdpSocket会发出readyRead（）信号。调用QUdpSocket :: readDatagram（）来读取数据报。

广播发送器和广播接收器示例显示如何使用Qt写入UDP发送器和UDP接收器。

QUdpSocket还支持多播。多播发送器和多播接收器示例显示如何使用写UDP多播客户端。

使用QHostInfo解析主机名

在建立网络连接之前，QTcpSocket和QUdpSocket执行名称查找，将您要连接的主机名转换为IP地址。通常使用DNS（域名服务）协议来执行此操作。

QHostInfo提供了一个静态函数，可让您自己执行此类查找。通过使用主机名，QObject指针和插槽签名调用QHostInfo :: lookupHost（），QHostInfo将执行名称查找并在结果准备好时调用给定的槽。实际的查找是在一个单独的线程中完成的，它利用操作系统自己的方法来执行名称查找。

QHostInfo还提供了一个名为QHostInfo :: fromName（）的静态函数，它将主机名作为参数并返回结果。在这种情况下，名称查找在与调用者相同的线程中执行。此重载对于非GUI应用程序或在单独的非GUI线程中进行名称查找非常有用。 （在GUI线程中调用此函数可能会导致用户界面冻结，而函数会在执行查找时阻塞。

支持网络代理

与Qt的网络通信可以通过代理执行，代理指导或过滤本地和远程连接之间的网络流量。

单个代理由QNetworkProxy类表示，该类用于描述和配置与代理的连接。支持在不同级别的网络通信上运行的代理类型，SOCKS 5支持允许代理低级别的网络流量，以及HTTP和FTP代理在协议级别工作。有关更多信息，请参阅QNetworkProxy :: ProxyType。

可以基于每个插槽或针对应用程序中的所有网络通信启用代理。通过在连接之前调用其QAbstractSocket :: setProxy（）函数，可以使新打开的套接字使用代理。通过使用QNetworkProxy :: setApplicationProxy（）函数，可以为所有后续套接字连接启用应用程序范围的代理。

代理工厂用于创建代理使用策略。 QNetworkProxyFactory根据特定代理类型的查询提供代理。查询本身在QNetworkProxyQuery对象中编码，这些对象允许根据关键标准选择代理，例如代理的目的（TCP，UDP，TCP服务器，URL请求），本地端口，远程主机和端口以及协议。使用（HTTP，FTP等）。

QNetworkProxyFactory :: proxyForQuery（）用于直接查询工厂。代理的应用程序范围策略可以通过将工厂传递给QNetworkProxyFactory :: setApplicationProxyFactory（）来实现，并且可以通过子类化QNetworkProxyFactory来创建自定义代理策略;有关详细信息，请参阅类文档。

承载管理支持

承载管理控制设备的连接状态，以便应用程序可以启动或停止网络接口，并在接入点之间透明地漫游。

QNetworkConfigurationManager类管理设备已知的网络配置列表。网络配置描述用于启动网络接口的参数集，由QNetworkConfiguration类表示。

通过基于给定网络配置打开QNetworkSession来启动网络接口。在大多数情况下，基于平台指定的默认网络配置创建网络会话是合适的。 QNetworkConfigurationManager :: defaultConfiguration（）函数返回默认网络配置。

在某些平台上，应用程序在执行任何网络操作之前​​打开网络会话是平台要求。这可以通过QNetworkConfigurationManager :: capabilities（）函数返回的值中的QNetworkConfigurationManager :: NetworkSessionRequired标志的呈现来测试。

另见承载管理。