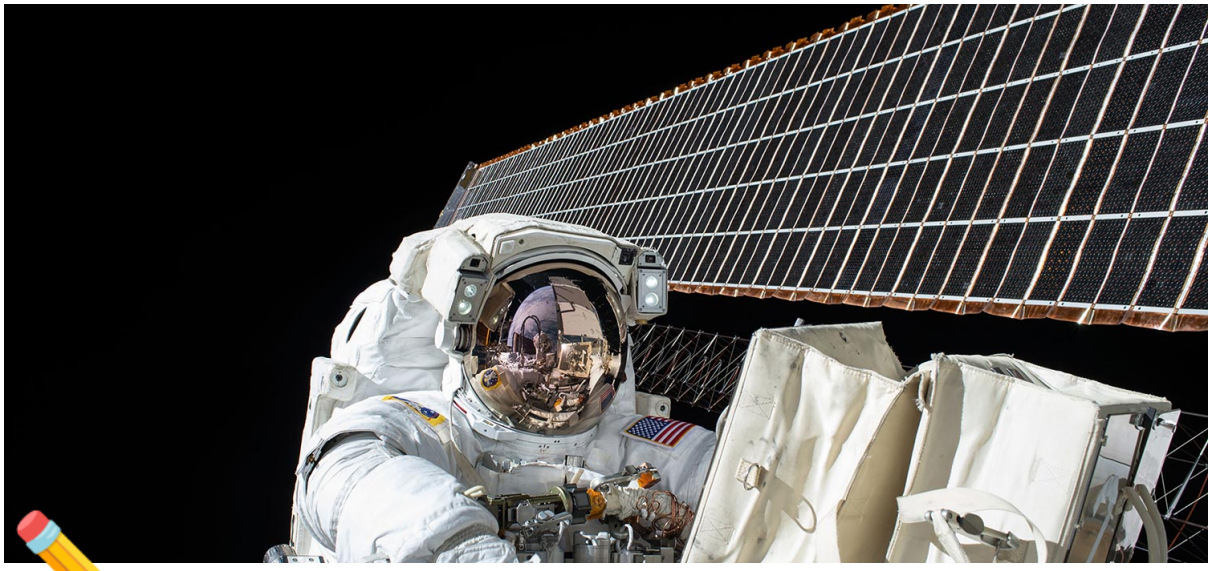


第27篇 (网络) HTTP

代码地址：https://github.com/Lornatang/QtStartQuicklyTutorial/tree/main/Network01

一、简单的网页浏览功能

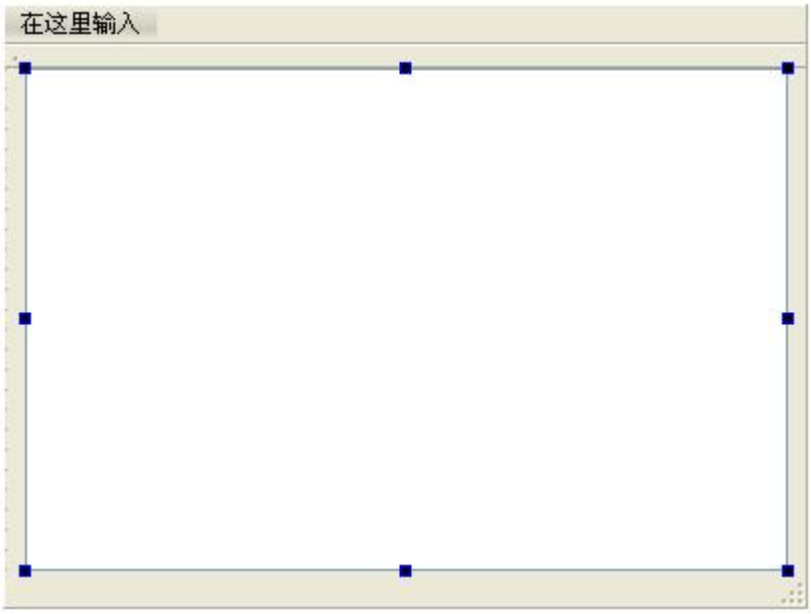
1．新建Qt Gui应用，项目名称为`http`，基类使用默认的`QMainWindow`即可，类名为`MainWindow`。

2．完成后打开`http.pro`文件，然后添加下面一行代码来使用网络模块：

```
QT  += network
```

然后保存该文件。

3．下面打开`mainwindow.ui`文件进入设计模式，向界面上添加一个`Text Browser`部件。效果如下图所示。



4．打开`mainwindow.h`文件，先包含头文件：`#include <QtNetwork>`

然后添加一个`private`私有对象定义：`QNetworkAccessManager *manager;`

最后添加一个私有槽声明：

```
private slots:
    void replyFinished(QNetworkReply *);
```

5．下面到`mainwindow.cpp`文件中，先在构造函数中添加如下代码：

```
manager = new QNetworkAccessManager(this);
connect(manager, SIGNAL(finished(QNetworkReply*)),
this,SLOT(replyFinished(QNetworkReply*)));
manager->get(QNetworkRequest(QUrl("http://www.qter.org")));
```

这里先创建了一个`QNetworkAccessManager`类的实例，它用来发送网络请求和接收应答。然后关联了管理器的`finished()`信号和我们自定义的槽，每当网络应答结束时都会发射这个信号。最后使用了`get()`函数来发送一个网络请求，网络请求使用`QNetworkRequest`类表示，`get()`函数返回一个`QNetworkReply`对象。除了`get()`函数，管理器还提供了发送HTTP POST请求的`post()`函数。

6．下面添加槽的定义：

```
void MainWindow::replyFinished(QNetworkReply *reply)
{
    QTextCodec *codec = QTextCodec::codecForName("utf8");
```

```
    QString all = codec->toUnicode(reply->readAll());
    ui->textBrowser->setText(all);
    reply->deleteLater();
}
```

因为`QNetworkReply`继承自`QIODevice`类，所以可以操作一般的I/O设备一样来操作该类。这里使用了`readAll()`函数来读取所有的应答数据，为了正常显示中文，使用了`QTextCodec`类来转换编码。在完成数据的读取后，需要使用`deleteLater()`来删除`replay`对象。

7．因为这里使用了`QtextCodec`类，所以还需要在`mainwindow.cpp`文件中包含头文件

```
#include <QTextCodec>
```

下面运行程序，效果如下图所示。

这里再将整个过程简答叙述一遍：上面实现了最简单的应用HTTP协议下载网页的功能。`QNetworkAccessManager`类用于发送网络请求和接受回复，具体来说，它是用`QNetworkRequest`类来管理请求，`QNetworkReply`类进行接收回复，并对数据进行处理。

在上面的代码中，我们使用了下面的代码来发送请求：

```
manager->get(QNetworkRequest(QUrl("http://www.qter.org")));
```

它返回一个`QNetworkReply`对象，这个后面再讲。我们只需知道只要发送请求成功，它就会下载数据。而当数据下载完成后，`manager`会发出`finished()`信号，我们关联了该信号：

```
connect(manager, SIGNAL(finished(QNetworkReply*)),
this,SLOT(replyFinished(QNetworkReply*)));
```

也就是说，当下载数据结束时，就会执行`replyFinished()`函数。在这个函数中我们对接收的数据进行处理：

```
QTextCodec *codec = QTextCodec::codecForName("utf8");
QString all = codec->toUnicode(reply->readAll());
ui->textBrowser->setText(all);
```

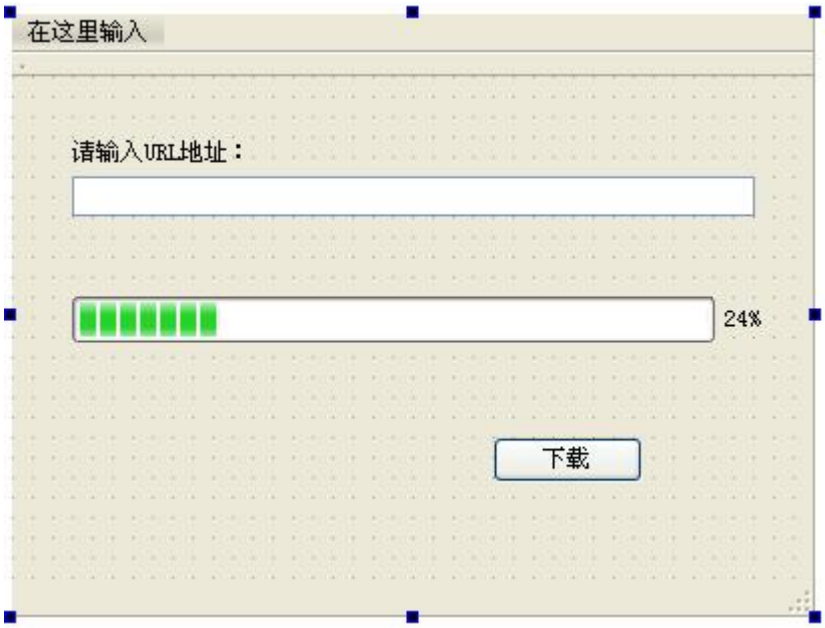
这里，为了能显示下载的网页中的中文，我们使用了`QTextCodec`类对象，应用utf8编码。使用`reply->readAll()`函数就可以将下载的所有数据读出。然后，我们在`textBrowser`中将数据显示出来。当`reply`对象已经完成了它的功能时，我们需要将它释放，就是最后一条代码：

```
reply->deleteLater();
```

二、实现下载文件功能

通过上面的例子可以看到，Qt中编写基于HTTP协议的程序是十分简单的，只有十几行代码。不过，一般我们下载文件都想要看到下载进度。下面我们就更改上面的程序，让它可以下载任意的文件，并且显示下载进度。

1．进入设计模式，删除以前的`Text Browser`部件，然后拖入一个`Line Edit`，一个`Label`，一个`Progress Bar`和一个`Push Button`，设计界面如下图所示。



2．在写代码之前，我们先介绍一下整个程序执行的流程：开始我们先让进度条隐藏。当我们在Line Edit中输入下载地址，点击下载按钮后，我们应用输入的下载地址，获得文件名，在磁盘上新建一个文件，用于保存下载的数据，然后进行链接，并显示进度条。在下载过程中，我们将每次获得的数据都写入文件中，并更新进度条，在接收完文件后，我们重新隐藏进度条，并做一些清理工作。根据这个思路，我们开始代码的编写。

3．到`mainwindow.h`中，首先添加`public`函数声明：

```
void startRequest(QUrl url); //请求链接
```

然后添加几个private变量和对象定义：

```
QNetworkReply *reply;
QUrl url;   //存储网络地址
QFile *file;  //文件指针
```

最后到`private slots`中，删除前面的`replyFinished(QNetworkReply *)`槽声明，并添加如下代码：

```
private slots:
    void on_pushButton_clicked();  //下载按钮的单击事件槽函数
    void httpFinished();  //完成下载后的处理
    void httpReadyRead();  //接收到数据时的处理
    void updateDataReadProgress(qint64, qint64); //更新进度条
```

4．下面到`mainwindow.cpp`文件中，将前面在构造函数中添加的内容删除，然后添加如下代码：

```
manager = new QNetworkAccessManager(this);
ui->progressBar->hide();
```

我们在构造函数中先隐藏进度条。等开始下载时再显示它。

5．下面将前面程序中添加的`replyFinished()`函数的定义删除，然后添加新的函数的定义。先添加网络请求函数的实现：

```
void MainWindow::startRequest(QUrl url)
{
    reply = manager->get(QNetworkRequest(url));
    connect(reply, SIGNAL(readyRead()), this, SLOT(httpReadyRead()));

    connect(reply, SIGNAL(downloadProgress(qint64, qint64)),
            this, SLOT(updateDataReadProgress(qint64, qint64)));

    connect(reply, SIGNAL(finished()), this, SLOT(httpFinished()));
}
```

这里使用了`get()`函数来发送网络请求，然后进行了`QNetworkReply`对象的几个信号和自定义槽的关联。其中`readyRead()`信号继承自`QIODevice`类，每当有新的数据可以读取时，都会发射该信号；每当网络请求的下载进度更新时都会发射`downloadProgress()`信号，它用来更新进度条；每当应答处理结束时，都会发射`finished()`信号，该信号与前面程序中`QNetworkAccessManager`类的`finished()`信号作用相同，只不过是发送者不同，参数也不同而已。下面添加几个槽的定义。

```
void MainWindow::httpReadyRead()
{
    if (file) file->write(reply->readAll());
}
```

这里先判断是否创建了文件，如果是，则读取返回的所有数据，然后写入到文件。该文件是在后面的“下载”按钮单击信号槽中创建并打开的。

```
void MainWindow::updateDataReadProgress(qint64 bytesRead, qint64 totalBytes)
{
    ui->progressBar->setMaximum(totalBytes);
    ui->progressBar->setValue(bytesRead);
}
```

这里设置了一下进度条的最大值和当前值。

```
void MainWindow::httpFinished()
{
    ui->progressBar->hide();
    file->flush();
    file->close();
    reply->deleteLater();
    reply = 0;
    delete file;
    file = 0;
}
```

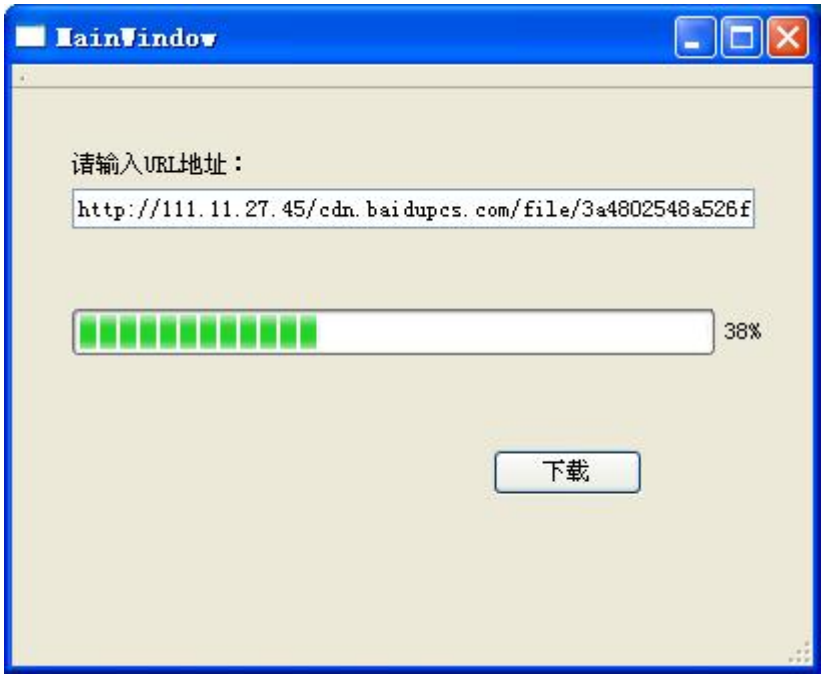
当完成下载后，重新隐藏进度条，然后删除`reply`和`file`对象。下面是“下载”按钮的单击信号的槽：

```
void MainWindow::on_pushButton_clicked()
{
    url = ui->lineEdit->text();
    QFileInfo info(url.path());
    QString fileName(info.fileName());
    if (fileName.isEmpty()) fileName = "index.html";
    file = new QFile(fileName);
    if(!file->open(QIODevice::WriteOnly))
    {
        qDebug() << "file open error";
        delete file;
        file = 0;
        return;
    }
    startRequest(url);
    ui->progressBar->setValue(0);
    ui->progressBar->show();
}
```

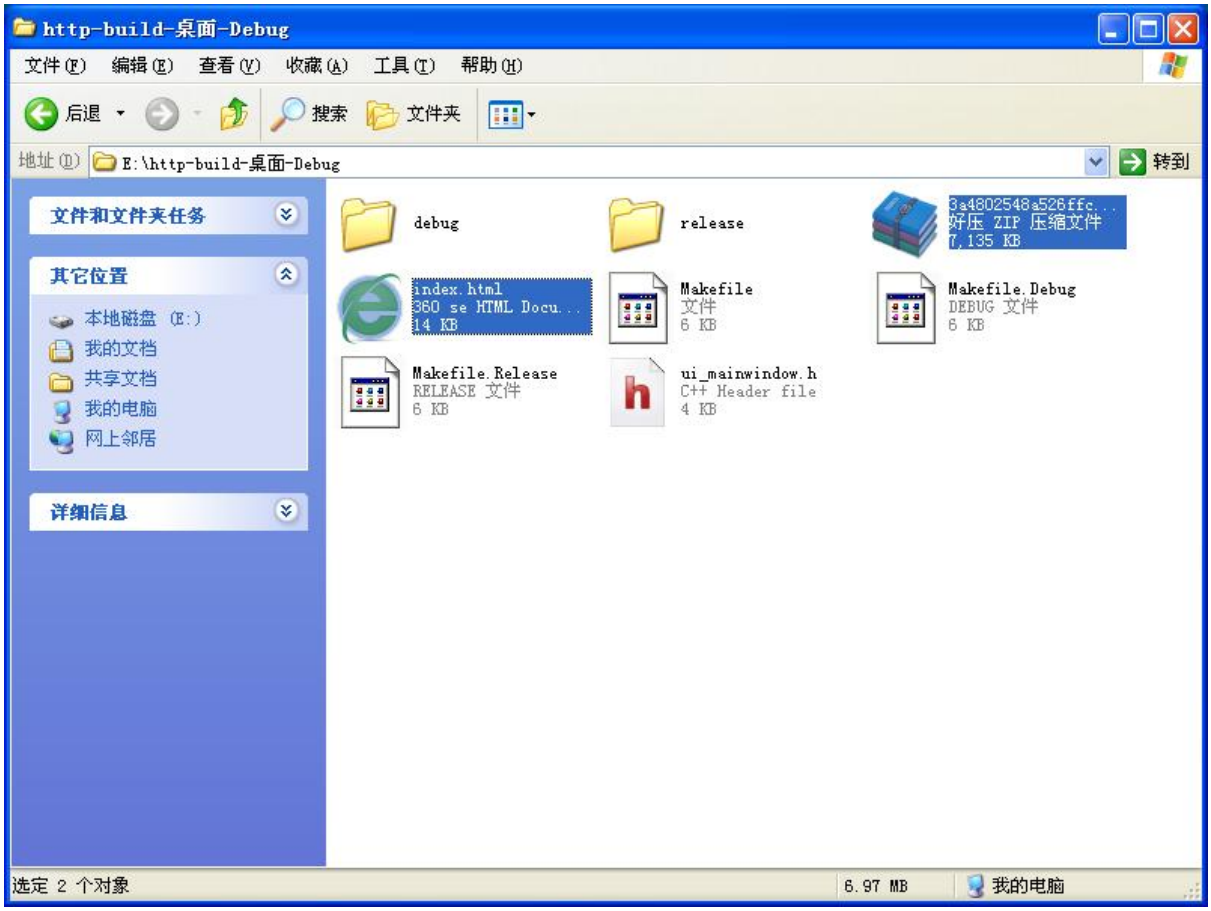
这里使用要下载的文件名创建了本地文件，然后使用输入的`url`进行了网络请求，并显示进度条。

6．下面运行程序，我们先输入`www.yafeilinux.com`的网址，下面一个网页。效果如下图所示。

完成后，可以尝试输入一个文件的下载地址，比如这里输入了《Qt Creator快速入门》一书在百度网盘上的地址，效果如下图所示。



7．最后，可以去项目编译生成的文件目录中查看下载的文件（我这里是`E:\http-build-桌面-Debug`），可以看到下载的文件，如下图所示。


http-build-桌面-Debug
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 文件夹
地址(D) E:\http-build-桌面-Debug
转到
文件和文件夹任务
其它位置
本地磁盘 (E:)
我的文档
共享文档
我的电脑
网上邻居
详细信息
debug
release
3a4802548a526ffc...
好压 ZIP 压缩文件
7,135 KB
index.html
360 se HTML Docu...
14 KB
Makefile
文件
6 KB
Makefile.Debug
DEBUG 文件
6 KB
Makefile.Release
RELEASE 文件
6 KB
ui_mainwindow.h
C++ Header file
4 KB
选定 2 个对象
6.97 MB
我的电脑