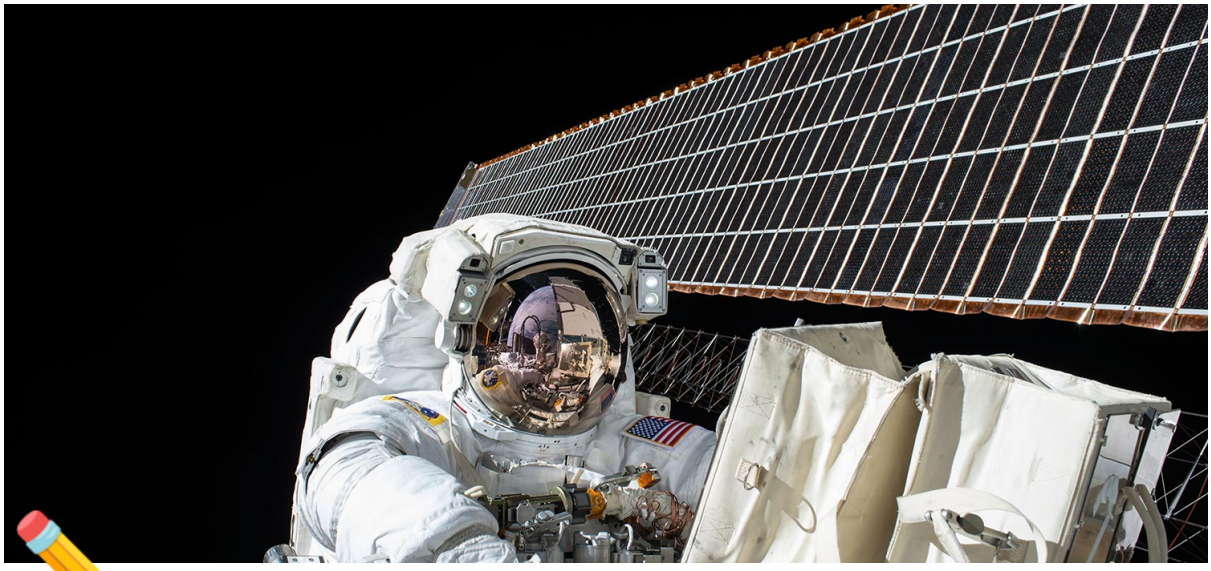
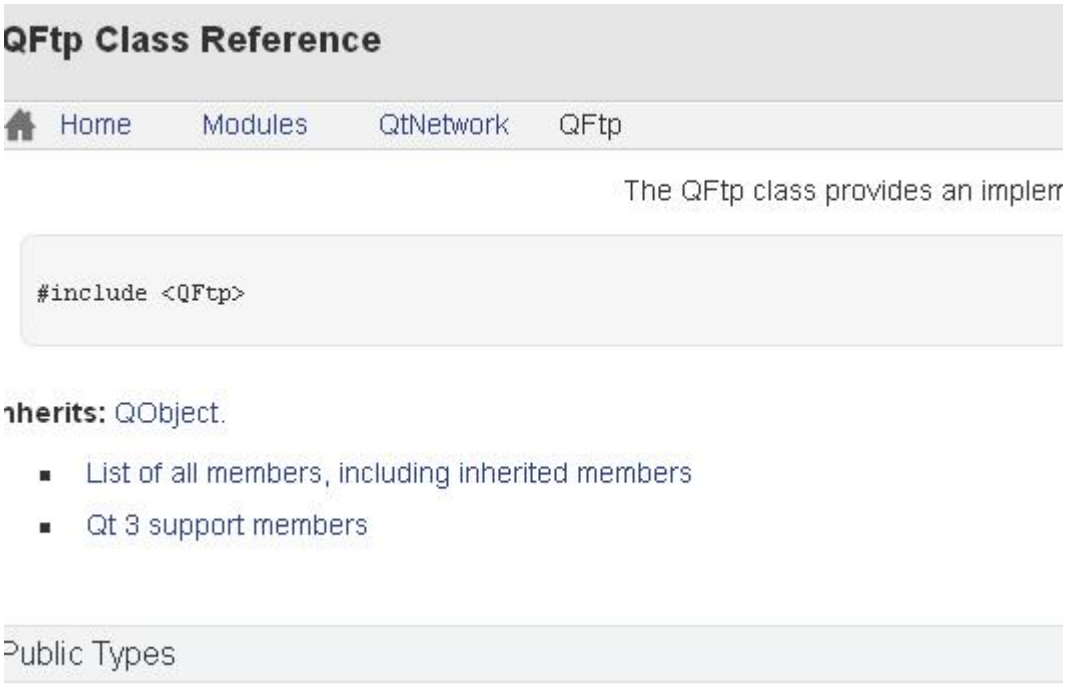


第28篇 (网络) FTP

一、简介

在Qt中，我们可以使用上一节讲述的`QNetworkAccessManager`和`QNetworkReply`类来进行FTP 程序的编写，因为它们用起来很简单。但是，对于较复杂的FTP操作，Qt还提供了`QFtp`类，利用这个类，我们很容易写出一个FTP客户端程序。下面我们先在帮助中查看这个类。



在QFtp中，所有的操作都对应一个特定的函数，我们可以称它们为命令。如`connectToHost()`连接到服务器命令，`login()`登录命令，`get()`下载命令，`mkdir()`新建目录命令等。因为`QFtp`类以异步方式工作，所以所有的这些函数都不是阻塞函数。也就是说，如果一个操作不能立即执行，那么这个函数就会直接返回，直到程序控制权返回Qt事件循环后才真正执行，它们不会影响界面的显示。

所有的命令都返回一个`int`型的编号，使用这个编号让我们可以跟踪这个命令，查看其执行状态。当每条命令开始执行时，都会发出`commandStarted()`信号，当该命令执行结束时，会发出`commandFinished()`信号。我们可以利用这两个信号和命令的编号来获取命令的执行状态。当然，如果不想执行每条命令都要记下它的编号，也可以使用`currentCommand()`来获取现在执行的命令，其返回值与命令的对应关系如下图。

enum QFtp::Command

This enum is used as the return value for the currentCommand() function. Thi
directory view when a list() command is started; in this case you can simply chec

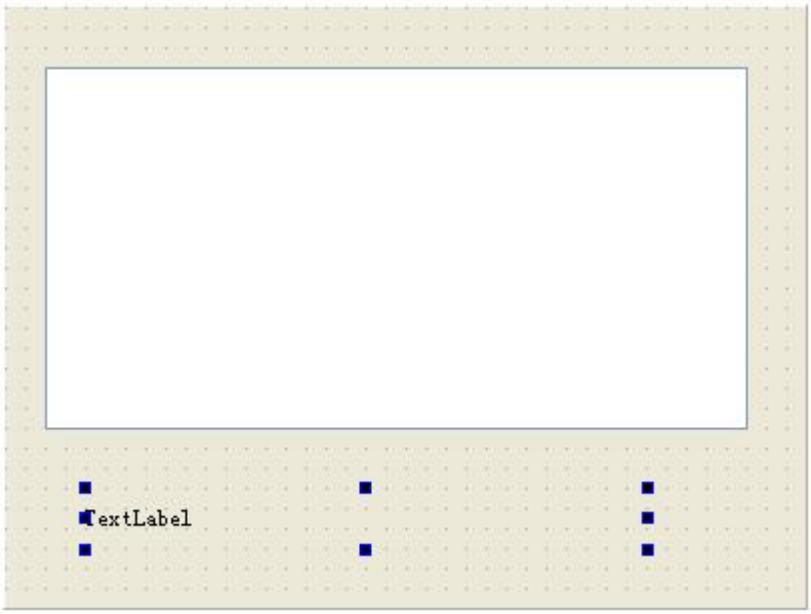
Constant	Value	Description
QFtp::None	0	No command is being executed.
QFtp::SetTransferMode	1	set the transfer mode.
QFtp::SetProxy	2	switch proxying on or off.
QFtp::ConnectToHost	3	connectToHost() is being executed.
QFtp::Login	4	login() is being executed.
QFtp::Close	5	close() is being executed.
QFtp::List	6	list() is being executed.
QFtp::Cd	7	cd() is being executed.
QFtp::Get	8	get() is being executed.
QFtp::Put	9	put() is being executed.
QFtp::Remove	10	remove() is being executed.
QFtp::Mkdir	11	mkdir() is being executed.
QFtp::Rmdir	12	rmdir() is being executed.
QFtp::Rename	13	rename() is being executed.
QFtp::RawCommand	14	rawCommand() is being executed.

二、实现简单的文件下载

下面我们先看一个简单的FTP客户端的例子，然后对它进行扩展。在这个例子中我们从FTP服务器上下载一个文件并显示出来。

1．我们新建Qt Gui应用。 项目名次为`myFtp`，基类选择`QWidget`，类名保持`Widget`即可。完成后打开`muFtp.pro`文件，在上面添加一行：`QT += network`，然后保存该文件。

2．修改`widget.ui`文件。 在其中添加一个`TextBrowser` 和一个 `Label`，效果如下。



3．在`main.cpp`中进行修改。

为了在程序中可以使用中文，我们在`main.cpp`中添加头文件`#include <QTextCodec>`

并在`main()`函数中添加代码：

```
QTextCodec::setCodecForTr(QTextCodec::codecForLocale());
```

4．在`widget.h`中进行修改。

先添加头文件：`#include <QFtp>`

再在`private`中定义对象：`QFtp *ftp;`

添加私有槽函数：

```
private slots:
    void ftpCommandStarted(int);
void ftpCommandFinished(int,bool);
```

5．在`widget.cpp`中进行更改。

（1）在构造函数中添加代码：

```
ftp = new QFtp(this);
ftp->connectToHost("ftp.qt-project.org"); //连接到服务器
ftp->login();   //登录
ftp->cd("qt/source");   //跳转到“qt”目录下的source目录中
ftp->get("INSTALL");  //下载“INSTALL”文件
ftp->close();   //关闭连接
```

```
// 当每条命令开始执行时发出相应的信号
connect(ftp,SIGNAL(commandStarted(int)),
        this,SLOT(ftpCommandStarted(int)));

// 当每条命令执行结束时发出相应的信号
connect(ftp,SIGNAL(commandFinished(int,bool)),
        this,SLOT(ftpCommandFinished(int,bool)));
```

我们在构造函数里执行了几个FTP的操作，登录站点，并下载了一个文件。然后又关联了两个信号和槽，用来跟踪命令的执行情况。

（2）实现槽函数：

```
void Widget::ftpCommandStarted(int)
{
    if(ftp->currentCommand() == QFtp::ConnectToHost){
       ui->label->setText(tr("正在连接到服务器..."));
    }
    if (ftp->currentCommand() == QFtp::Login){
       ui->label->setText(tr("正在登录..."));
    }
    if (ftp->currentCommand() == QFtp::Get){
       ui->label->setText(tr("正在下载..."));
    }
    else if (ftp->currentCommand() == QFtp::Close){
       ui->label->setText(tr("正在关闭连接..."));
    }
}
```

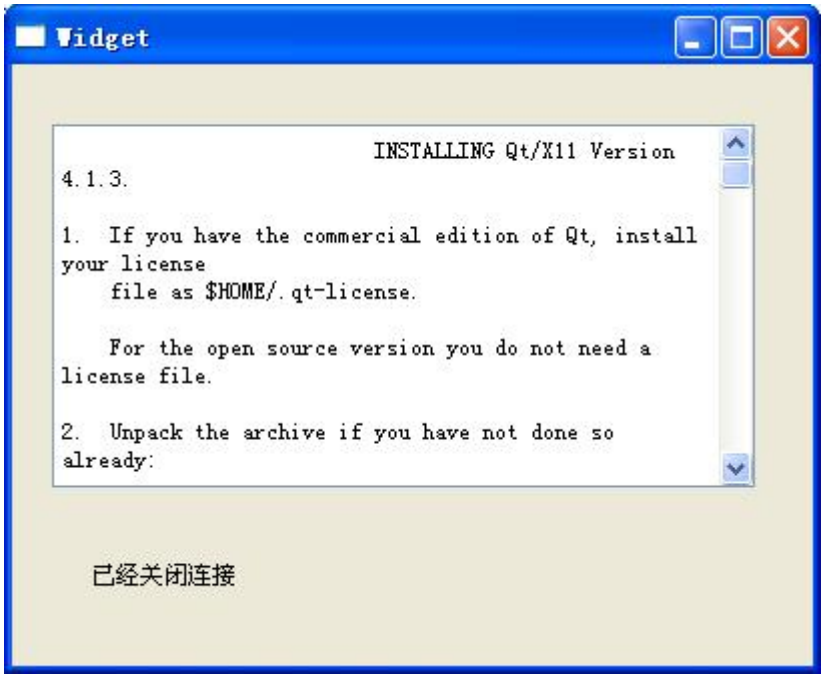
每当命令执行时，都会执行`ftpCommandStarted()`函数，它有一个参数`int id`，这个`id`就是调用命令时返回的`id`，如`int loginID= ftp->login();`这时，我们就可以用`if(id == loginID)`来判断执行的是否是`login()`函数。但是，我们不想为每个命令都设置一个变量来存储其返回值，所以，我们这里使用了`ftp->currentCommand()`，它也能获取当前执行的命令的类型。在这个函数里我们让开始不同的命令时显示不同的状态信息。

```
void Widget::ftpCommandFinished(int,bool error)
{
    if(ftp->currentCommand() == QFtp::ConnectToHost){
       if(error)
           ui->label->setText(tr("连接服务器出现错误：%1")
                              .arg(ftp->errorString()));
       else ui->label->setText(tr("连接到服务器成功"));
    }
    if (ftp->currentCommand() == QFtp::Login){
       if(error)
           ui->label->setText(tr("登录出现错误：%1")
                              .arg(ftp->errorString()));
       else ui->label->setText(tr("登录成功"));
    }
```

```
    if (ftp->currentCommand() == QFtp::Get){
       if(error)
           ui->label->setText(tr("下载出现错误：%1")
                              .arg(ftp->errorString()));
       else {
           ui->label->setText(tr("已经完成下载"));
           ui->textBrowser->setText(ftp->readAll());
       }
    }
    else if (ftp->currentCommand() == QFtp::Close){
       ui->label->setText(tr("已经关闭连接"));
    }
}
```

这个函数与`ftpCommandStarted()`函数相似，但是，它是在一个命令执行结束时执行的。它有两个参数，第一个`intid`，就是调用命令时返回的编号，我们在上面已经讲过了。第二个是`bool error`，它标志现在执行的命令是否出现了错误。如果出现了错误，那么`error`为`true`，否则为`false`。我们可以利用它来输出错误信息。在这个函数中，我们在完成一条命令时显示不同的状态信息，并显示可能的出错信息。在`if (ftp->currentCommand() == QFtp::Get)`中，也就是已经完成下载时，我们让`textBrowser`显示下载的信息。

6．运行程序，效果如下。



7．出错演示。

下面我们演示一下出错时的情况。

将构造函数中的代码`ftp->login();`改为`ftp->login("tom","123456");`

这时我们再运行程序：

可以看到，它输出了错误信息，指明了错误的指令和出错的内容。其实我们设置的这个错误，也是想告诉大家，在FTP中如果没有设置用户名和密码，那么默认的用户名应该是`anonymous`，这时密码可以任意填写，而使用其他用户名是会出错的。

三、修改界面

我们删除了`TextBrowser`，加入了几个`Label`，`Line Edit`，`Push Button`部件，一个`Tree Widge`t及一个`Progress Bar`部件。然后我们对其中几个部件做如下更改。

（1）将“FTP服务器”标签后的`Line Edit`的`objectName`属性改为`ftpServerLineEdit`，其`text`属性改为`ftp.qt-project.org`。

（2）将“用户名”标签后的`Line Edit`的`objectName`属性改为`userNameLineEdit`，其`text`属性改为`anonymous`，将其`toolTip`属性改为“默认用户名请使用：anonymous，此时密码任意。”

（3）将“密码”标签后的`Line Edit`的`objectName`属性改为`passWordLineEdit`，其`text`属性改为`123456`，将其`echoMode`属性改为`Password`。

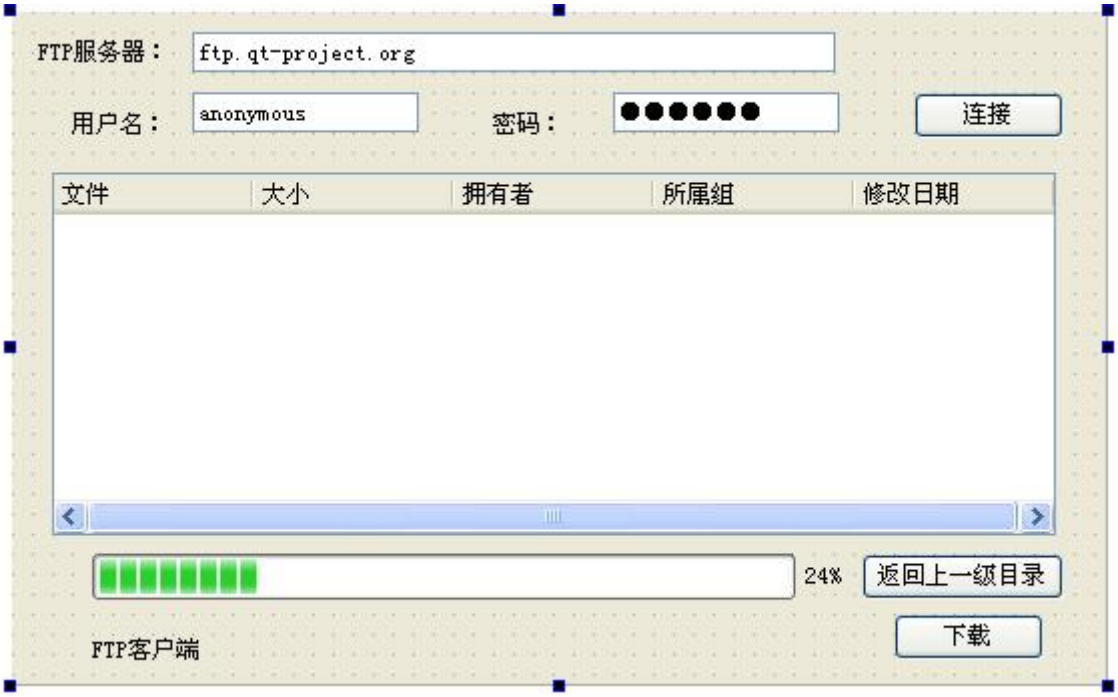
（4）将“连接”按钮的`objectName`属性改为`connectButton`。

（5）将“返回上一级目录”按钮的`objectName`属性改为`cdToParentButton`。

（6）将“下载”按钮`的objectName`属性改为`downloadButton`。

（7）将`Tree Widget`的`objectName`属性改为`fileList`，然后在`Tree Widget`部件上单击鼠标右键，选择`Edit Items`菜单，添加列属性如下。

最终界面如图所示：



下面我们的程序中，就是实现在用户填写完相关信息后，按下“连接”按钮，就可以连接到FTP服务器，并在`TreeWidget`中显示服务器上的所有文件，我们可以按下“下载”按钮来下载选中的文件，并使用进度条显示下载进度。

四、功能实现

1．更改`widget.h`文件。

（1）添加头文件`#include <QtGui>`

（2）在`private`中添加变量：

```
QHash<QString, bool> isDirectory; //用来存储一个路径是否为目录的信息
QString currentPath; //用来存储现在的路径
QFile *file;
```

（3）添加槽：

```
private slots:
void on_downloadButton_clicked();
void on_cdToParentButton_clicked();
void on_connectButton_clicked();
void ftpCommandFinished(int,bool);
void ftpCommandStarted(int);
void updateDataTransferProgress(qint64,qint64 );//更新进度条
//将服务器上的文件添加到Tree Widget中
void addToList(const QUrlInfo &urlInfo);
void processItem(QTreeWidgetItem*,int);//双击一个目录时显示其内容
```

2．更改`widget.cpp`的内容。

（1）实现“连接”按钮的单击事件槽。

```
void Widget::on_connectButton_clicked()   //连接按钮
{
    ui->fileList->clear();
    currentPath.clear();
    isDirectory.clear();

    ftp = new QFtp(this);
    connect(ftp,SIGNAL(commandStarted(int)),
this,SLOT(ftpCommandStarted(int)));
    connect(ftp,SIGNAL(commandFinished(int,bool)),
           this,SLOT(ftpCommandFinished(int,bool)));
    connect(ftp,SIGNAL(listInfo(QUrlInfo)),
this,SLOT(addToList(QUrlInfo)));
    connect(ftp,SIGNAL(dataTransferProgress(qint64,qint64)),
           this,SLOT(updateDataTransferProgress(qint64,qint64)));

    QString ftpServer = ui->ftpServerLineEdit->text();
    QString userName = ui->userNameLineEdit->text();
    QString passWord = ui->passWordLineEdit->text();
    ftp->connectToHost(ftpServer,21); //连接到服务器,默认端口号是21
```

```
    ftp->login(userName,passWord);   //登录
}
```

我们在“连接”按钮的单击事件槽函数中新建了`ftp`对象，然后关联了相关的信号和槽。这里的`listInfo()`信号由`ftp->list()`函数发射，它将在登录命令完成时调用，下面我们提到。而`dataTransferProgress()`信号在数据传输时自动发射。最后我们从界面上获得服务器地址，用户名和密码等信息，并以它们为参数执行连接和登录命令。

（2）更改`ftpCommandFinished()`函数。

我们在相应位置做更改。

首先，在登录命令完成时，我们调用list()函数：

```
ui->label->setText(tr("登录成功"));
ftp->list();   //发射listInfo()信号，显示文件列表
然后，在下载命令完成时，我们使下载按钮可用，并关闭打开的文件。
ui->label->setText(tr("已经完成下载"));
ui->downloadButton->setEnabled(true);
file->close();
delete file;
```

最后再添加一个`if`语句，处理`list`命令完成时的情况：

```
if (ftp->currentCommand() == QFtp::List){
       if (isDirectory.isEmpty())
       { //如果目录为空,显示“empty”
           ui->fileList->addTopLevelItem(
                  new QTreeWidgetItem(QStringList()<< tr("<empty>")));
           ui->fileList->setEnabled(false);
           ui->label->setText(tr("该目录为空"));
       }
}
```

我们在`list`命令完成时，判断文件列表是否为空，如果为空，就让`Tree Widget`不可用，并显示“empty”条目。

（3）添加文件列表函数的内容如下。

```
void Widget::addToList(const QUrlInfo &urlInfo)  //添加文件列表
{
    QTreeWidgetItem *item = new QTreeWidgetItem;
    item->setText(0, urlInfo.name());
    item->setText(1, QString::number(urlInfo.size()));
    item->setText(2, urlInfo.owner());
    item->setText(3, urlInfo.group());
    item->setText(4, urlInfo.lastModified().toString("MMM dd yyyy"));
```

```
    QPixmap pixmap(urlInfo.isDir() ? "../myFtp2/dir.png" : "../myFtp2/file.png");
    item->setIcon(0, pixmap);

    isDirectory[urlInfo.name()] = urlInfo.isDir();
    //存储该路径是否为目录的信息
    ui->fileList->addTopLevelItem(item);
    if (!ui->fileList->currentItem()) {
       ui->fileList->setCurrentItem(ui->fileList->topLevelItem(0));
       ui->fileList->setEnabled(true);
    }
}
```

当`ftp->list()`函数执行时会发射`listInfo()`信号，此时就会执行`addToList()`函数，在这里我们将文件信息显示在`Tree Widget`上，并在`isDirectory`中存储该文件的路径及其是否为目录的信息。为了使文件与目录进行区分，我们使用了不同的图标`file.png`和`dir.png`来表示它们，这两个图标放在了工程文件夹中。

（4）将构造函数的内容更改如下。

```
{
    ui->setupUi(this);
    ui->progressBar->setValue(0);
    //鼠标双击列表中的目录时，我们进入该目录
    connect(ui->fileList,SIGNAL(itemActivated(QTreeWidgetItem*,int)),
           this,SLOT(processItem(QTreeWidgetItem*,int)));
}
```

这里我们只是让进度条的值为0，然后关联了`Tree Widget`的一个信号`itemActivated()`。当鼠标双击一个条目时，发射该信号，我们在槽函数中判断该条目是否为目录，如果是则进入该目录。

（5）`processItem()`函数的实现如下。

```
void Widget::processItem(QTreeWidgetItem* item,int)  //打开一个目录
{
    QString name = item->text(0);
    if (isDirectory.value(name)) {  //如果这个文件是个目录，则打开
       ui->fileList->clear();
       isDirectory.clear();
       currentPath += '/';
       currentPath += name;
       ftp->cd(name);
       ftp->list();
       ui->cdToParentButton->setEnabled(true);
    }
}
```

（6）“返回上一级目录”按钮的单击事件槽函数如下。

```
void Widget::on_cdToParentButton_clicked()  //返回上级目录按钮
{
    ui->fileList->clear();
    isDirectory.clear();
    currentPath = currentPath.left(currentPath.lastIndexOf('/'));
    if (currentPath.isEmpty()) {
       ui->cdToParentButton->setEnabled(false);
       ftp->cd("/");
    } else {
       ftp->cd(currentPath);
    }
    ftp->list();
}
```

在返回上一级目录时，我们取当前路径的最后一个`/`之前的部分，如果此时路径为空了，我们就让“返回上一级目录”按钮不可用。

（7）“下载”按钮单击事件槽函数如下。

```
void Widget::on_downloadButton_clicked()  //下载按钮
{
    QString fileName = ui->fileList->currentItem()->text(0);
    file = new QFile(fileName);
    if (!file->open(QIODevice::WriteOnly))
    {
       delete file;
       return;
}
//下载按钮不可用，等下载完成后才可用
    ui->downloadButton->setEnabled(false);    ftp->get(ui->fileList->currentItem()->te
xt(0), file);
}
```

在这里我们获取了当前项目的文件名，然后新建文件，使用`get()`命令下载服务器上的文件到我们新建的文件中。

（8）更新进度条函数内容如下。

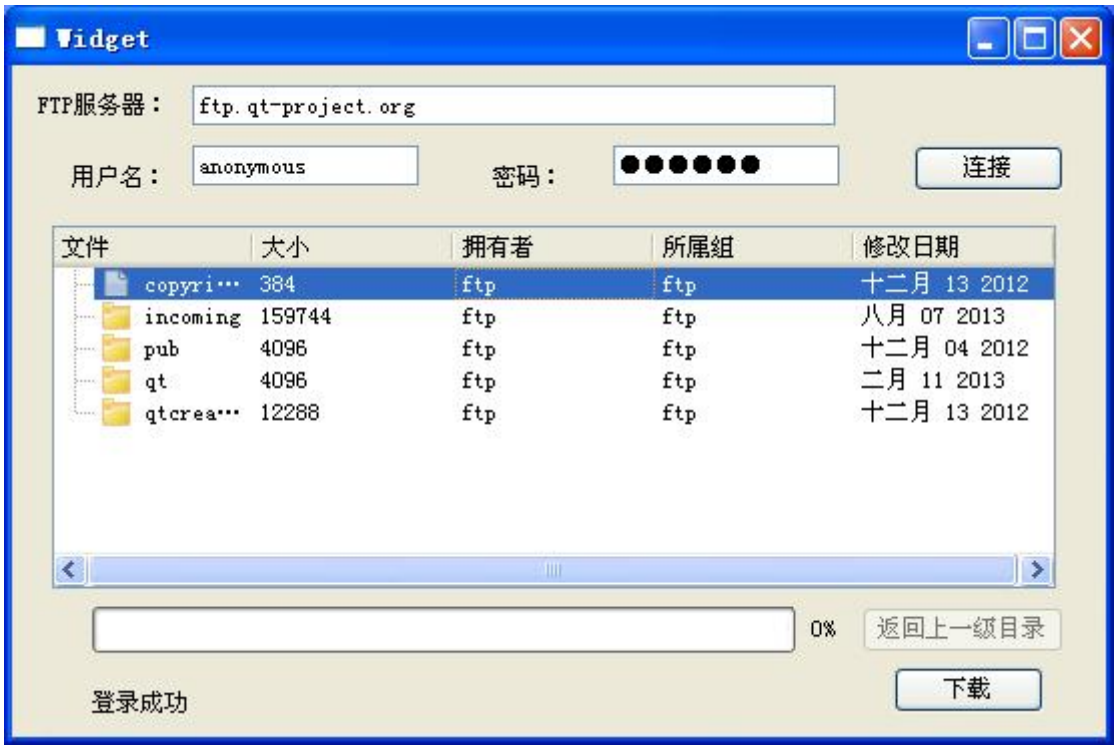
```
void Widget::updateDataTransferProgress( //进度条
                  qint64 readBytes,qint64 totalBytes)
{
    ui->progressBar->setMaximum(totalBytes);
    ui->progressBar->setValue(readBytes);
}
```

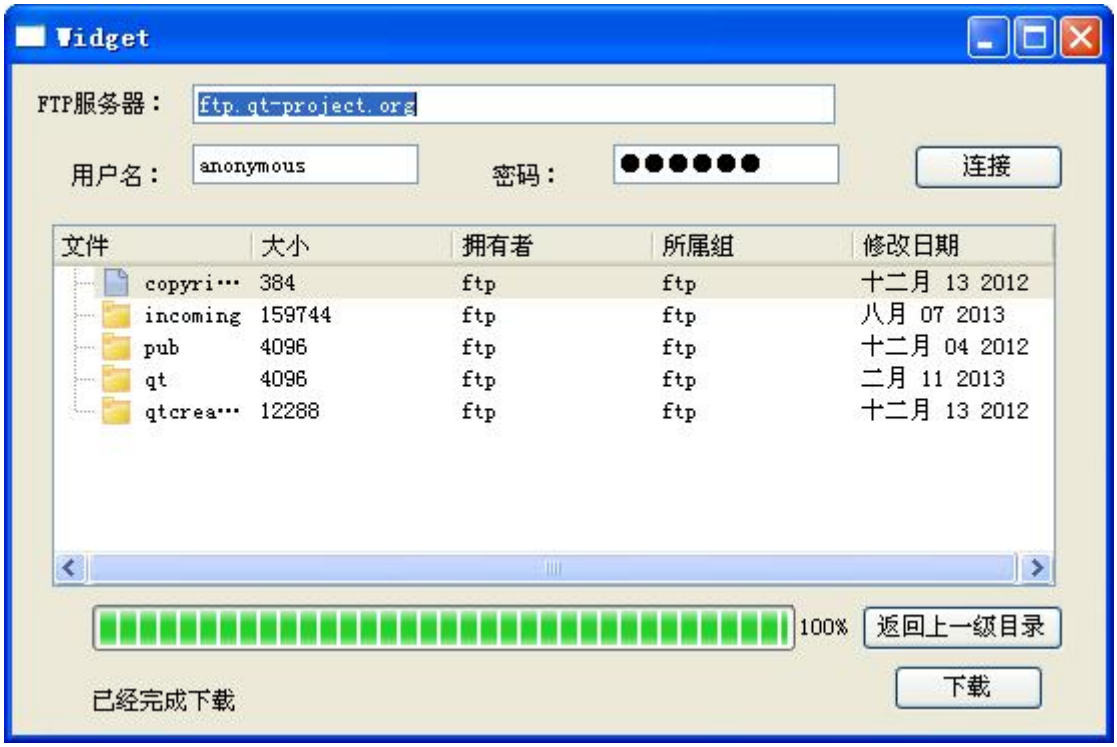
3．流程说明。

整个程序的流程就和我们实现函数的顺序一样。用户在界面上输入服务器的相关信息，然后我们利用这些信息进行连接并登录服务器，等登录服务器成功时，我们列出服务器上所有的文件。对于一个目录，我们可以进入其中，并返回上一级目录，我们可以下载文件，并显示下载的进度。

对于`ftp`的操作，全部由那些命令和信号来完成，我们只需要调用相应的命令，并在其发出信号时，进行对应的处理就可以了。而对于文件的显示，则是视图部分的知识了。

4．运行程序，效果如下图所示。




Widget
FTP服务器：
ftp.qt-project.org
用户名：
anonymous
密码：
连接
文件
大小
拥有者
所属組
修改日期
copyri…
384
ftp
ftp
十二月 13 2012
incoming
159744
ftp
ftp
八月 07 2013
pub
4096
ftp
ftp
十二月 04 2012
qt
4096
ftp
ftp
二月 11 2013
qtcrea…
12288
ftp
ftp
十二月 13 2012
100%
返回上一级目录
已经完成下载
下载