بسم الله الرحمن الرحيم

الشبكات في Qt

ط/ محمد العبدلي

يسرني أن أهدي هذا المؤلف الى والدتي .

الشبكات :

مقدمة:

تدعم مكتة Qt الإتصال بجميع طرقه ووسائله فهي توفر لك صفوفة للإتصال ب tcp و udp و جميع مايتفرع منها من بروتوكولات أخرى .
بالإضافة الى ذلك فهي تدعم العميل بتوفير بروتوكولات مثل ftp و http وجميع هذه الخدمات سوف تجدها متوفرة على QtNetwork .

كيف تدمجه داخل عملك :

جميع صفوف الشبكات هي جزء من وحدة QtNetwork وهي مدمجة مع المصادر المفتوحة فقط عليك ان تظيف العبارة QT += network داخل ملف المشروع.

التعامل مع بروتوكولات العميل :

هناك صفوف مثل QFtp و QHttp وهي تقوم بتغليف كل من البروتوكولين http و ftp
وهي سهلة الإستخدام لانها ذات مستوى عالي في التعامل ويمكنك صنع مثيلات لها بدون الإعتماد عليهما بإستخدام tcp server و socket فهي تعتبر ذات مستوى أدني من سابقيه وسوف نأتي في الحديث عنهم في وقت لاحق من نفس الفصل .
ويتمثل الفرق بين FTP و HTTP ان الأول بروتوكول نقل الملفات والأخير بروتوكول لنقل النصوص التشعبية و من ثم الأول اكثر تعقيدا في التعامل مع الأخير لأنه يعتمد على من المتصل واتصال الأوامر ببعضها اما الأخير فهو ينفذ كل طلبية منفصلة عن الطلبية الأخرى ولكن في نهاية الموضوع انت تريد ان تعرف كيف تستخدمهم لتطوير أعمالك؟؟.

التعامل مع ftp (بروتوكول نقل الملفات) :

كما ذكرنا سابقا ان الصف الذي يتعامل مع هذا البروتوكول هو QFtp ليقوم بإستقبال عناوين ftp://ftp.example.net ليقوم بارسال واستقبال الملفات من خلالها وبالتأكيد هو يحتاج الى اسم مستخدم و كلمة المرور لكي يستطيع ان يقوم بالعملية السابقة.

أهم الدوال المستخدمة :

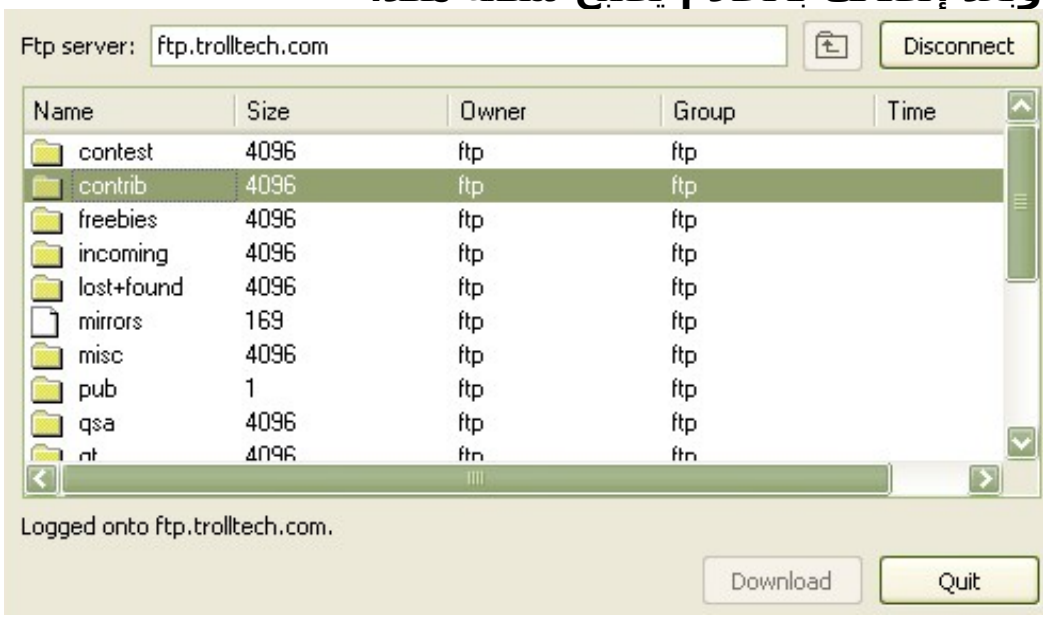
دوال الإتصال
connectToHost عملية الإتصال بالخادم وتستقبل في وسيطها الأول اسم الخادم والآخر المنفذ.
login عملية الدخول الى الملفات لتمكنك من التحميل والرفع من وإلى الخادم وتستقبل في وسيط الأول المستخدم والثاني كلمة المرور .
دوال تحميل ورفع الملفات
Get لتحميل الملفات من الخادم .
Put لرفع الملفات الى الخادم .
دوال التحكم بالمجلدات والملفات
List ويعيد قائمة من الملفات الموجودة في المسار الحالي.
Mkdir ويقوم بصنع مجلد جديد .
Rmdir حذف مجلد.
Remove حذف ملف.
Rename اعادة تسمية ملف.
دوال التحكم بالأوامر :
الأوامر :جميع الدوال التي ذكرناها سابقا هي اوامر يتم ارسالها الى الخادم لينفذها بعد ذلك بالتدريج.
منها ماهو اشارات يتم ارسالها مثل
commsndFinished ويتم ارسالها عند الإنتهاء من أمر ما .
commandStarted ويتم ارسالها عند البدء بأمر جديد .
done ويتم إرسالها عند الإنتهاء من جميع الأوامر المجدولة .
ومنها ماهو مستقبلات مثل :
Abort ويقوم بالخروج من الأمر الذي يتم تنفيذه في هذه اللحظة والغاء جميع الأوامر المجدولة .
ومنها ماهو دوال عامة مثل :
currentId ويعيد رقم الأمر اذي يجري تنفيذه الآن .
currentCommand ويعيد لك نوع الإجراء الحالي ويكون من النوع الثوابت المرقمة ل command
دوال أخرى :
read و readAll وتقرأ جميع البيانات القادمة عن طريق ال socket وبالـأكيد في هذه الحالة انت تحتاج الى اشارة readyRead لكي تتأكد من توفر بيانات للقراءة.
للوصول للمستوى المنخفض في Ftp انت بحاجة الى rawCommand لترسل اوامرك
ولكن في النهاية الأوامر السابقة هي الأسهل والأكثر أمانا .

برنامج نقل الملفات :

سوف نقوم الآن بتطبيق مثال على ما تطرقنا له من حديث عن بروتوكول ftp بإستخدام Qt

في هذا المثال سوف تستطيع الإتصال بالخادم ليسرد لك جميع الملفات التي به ليمكنك من تحميلها ان اردت .
ملاحظة: المثال متوفر في اداة examples المرف مع Qt

وبعد إتصالك بالخادم يصبح شكله هكذا



رأس الصف

```
1. #ifndef FTPWINDOW_H
2. #define FTPWINDOW_H
3. #include <QtGui>
4. #include <QtNetwork>
5. class FtpWindow : public QDialog
6. {
7. Q_OBJECT
8. public:
```

```
9. FtpWindow(QWidget *parent = 0);
10.QSize sizeHint() const;
11.private slots:
12.void connectOrDisconnect();
13.void downloadFile();
14.void cancelDownload();
15.void ftpCommandFinished(int commandId, bool error);
16.void addToList(const QUrlInfo &urlInfo);
17.void processItem(QTreeWidgetItem *item, int column);
18.void cdToParent();
19.void updateDataTransferProgress(qint64
   readBytes,qint64 totalBytes);
20.void enableDownloadButton();
21.private:
22.QLabel *ftpServerLabel;
23.QLineEdit *ftpServerLineEdit;
24.QLabel *statusLabel;
25.QTreeWidget *fileList;
26.QPushButton *cdToParentButton;
27.QPushButton *connectButton;
28.QPushButton *downloadButton;
29.QPushButton *quitButton;
30.QDialogButtonBox *buttonBox;
31.QProgressDialog *progressDialog;
32.QHash<QString, bool> isDirectory;
33.QString currentPath;
34.QFtp *ftp;
35.QFile *file;
36.};
37.#endif
```

في هذا الصف قمنا بصناعة دوال استقبال الإشارات مثل الإتصال وقطع الإتصال و طلب التحميل و إلغاء التحميل و مؤشرات لمتغيرات
مثل **ftp** في السطر **37** تتبعها لأنها الأهم .

جسم الصف

دالة البناء

```
1. FtpWindow::FtpWindow(QWidget *parent)
2. : QDialog(parent), ftp(0)
3. {
4. ftpServerLabel = new QLabel(tr("Ftp &server:"));
5. ftpServerLineEdit = new QLineEdit("ftp.trolltech.com");
6. ftpServerLabel->setBuddy(ftpServerLineEdit);
7. statusLabel = new QLabel(tr("Please enter the name of an FTP
   server."));
8. fileList = new QTreeWidget;
9. fileList->setEnabled(false);
10.fileList->setRootIsDecorated(false);
   11.fileList->setHeaderLabels(QStringList() << tr("Name") <<
     tr("Size") << tr("Owner") << tr("Group") << tr("Time"));
```

```
12.fileList->header()->setStretchLastSection(false);
13.connectButton = new QPushButton(tr("Connect"));
14.connectButton->setDefault(true);
15.cdToParentButton = new QPushButton;
16.cdToParentButton-
  >setIcon(QPixmap(":/images/cdtoparent.png"));
17.cdToParentButton->setEnabled(false);
18.downloadButton = new QPushButton(tr("Download"));
19.downloadButton->setEnabled(false);
20.quitButton = new QPushButton(tr("Quit"));
21.buttonBox = new QDialogButtonBox;
22.buttonBox->addButton(downloadButton,
  QDialogButtonBox::ActionRole);
23.buttonBox->addButton(quitButton,
  QDialogButtonBox::RejectRole);
24.progressDialog = new QProgressDialog(this);
25.connect(fileList, SIGNAL(itemActivated(QTreeWidgetItem *,
  int)),
26.this, SLOT(processItem(QTreeWidgetItem *, int)));
27.connect(fileList, SIGNAL(currentItemChanged(QTreeWidgetItem
  *, QTreeWidgetItem*)),
28.this, SLOT(enableDownloadButton()));
29.connect(progressDialog, SIGNAL(canceled()), this,
  SLOT(cancelDownload()));
30.connect(connectButton, SIGNAL(clicked()), this,
  SLOT(connectOrDisconnect()));
31.connect(cdToParentButton, SIGNAL(clicked()), this,
  SLOT(cdToParent()));
32.connect(downloadButton, SIGNAL(clicked()), this,
  SLOT(downloadFile()));
33.connect(quitButton, SIGNAL(clicked()), this, SLOT(close()));
34.QHBoxLayout *topLayout = new QHBoxLayout;
35.topLayout->addWidget(ftpServerLabel);
36.topLayout->addWidget(ftpServerLineEdit);
37.topLayout->addWidget(cdToParentButton);
38.topLayout->addWidget(connectButton);
39.QVBoxLayout *mainLayout = new QVBoxLayout;
40.mainLayout->addLayout(topLayout);
41.mainLayout->addWidget(fileList);
42.mainLayout->addWidget(statusLabel);
43.mainLayout->addWidget(buttonBox);
44.setLayout(mainLayout);
45.setWindowTitle(tr("FTP"));
46.}
```

في بادئ الأمر قمنا بتهيئة المتغيرات في الأسطر من 1 الى 5 .
وبعد ذلك قمنا بالربط بين ftpServerLabel و
ftpServerLineEdit بإستخدام setBuddy في السطر رقم 6 , وفي السطرين 7و 8 قمنا بتهيئة المتغيرين statusLabel و fileList

والأخير سوف يستقبل جميع أسماء الملفات في الخادم ووضعنه في حالة مقفل كبداية الى ان يتم الإتصال بالخادم ثم بعد ذلك قمنا بتهيئة زر الإتصال وربطناه بزر **enter** في الأسطر **13** و **14** ومن بعدها هيأنا
زر الرجوع الى المجلد الأب وأضفنا له أيقونة وجعلناه مقفل حتى يتم الإتصال في الأسطر من **15** الى **17** ومن ثم قمنا بتهيئة زر تحميل الملف وجعلناه في حالة مقفلة الى ان يتم الإتصال بالخادم في السطرين **18** و **19** وبعدها هيأنا زر الخروج من البرنامج وصندوق الأزرار وأضفنا للصندوق كلا من زري التحميل وزر الخروج الأول يبعث اشارة الموافقة والثاني اشارة الرفض في الأسطر من **20** الى **23** ثم صنعنا صندوق شريط التحميل في السطر **24** ومنها الى **33** هي اسطر ربط الإشارات مع المستقبلات بواسطة **connect** ومنها الى السطر **44** قمنا بترتيب العناصر التي صنعناها داخل **layouts** ومن ثم اعطينا عنوان للنافذة لنغلق على دالة البناء .

```
1. QSize FtpWindow::sizeHint() const
2. {
3. return QSize(500, 300);
4. }
```

لتحديد مساحة النافذة .
الدالة الأكثر أهمية

```
1. void FtpWindow::connectOrDisconnect()
2. {
3. if (ftp) {
4. ftp->abort();
5. ftp->deleteLater();
6. ftp = 0;
7. fileList->setEnabled(false);
8. cdToParentButton->setEnabled(false);
9. downloadButton->setEnabled(false);
10.connectButton->setEnabled(true);
11.connectButton->setText(tr("Connect"));
12.#ifndef QT_NO_CURSOR
13.setCursor(Qt::ArrowCursor);
14.#endif
15.return;
16.}
17.#ifndef QT_NO_CURSOR
18.setCursor(Qt::WaitCursor);
19.#endif
20.ftp = new QFtp(this);
21.connect(ftp, SIGNAL(commandFinished(int, bool)),
22.this, SLOT(ftpCommandFinished(int, bool)));
23.connect(ftp, SIGNAL(listInfo(const QUrlInfo &)),
24.this, SLOT(addToList(const QUrlInfo &)));
25.connect(ftp, SIGNAL(dataTransferProgress(qint64,
   qint64)),
```

```
this, SLOT(updateDataTransferProgress(qint64,
  qint64)));
fileList->clear();
currentPath.clear();
isDirectory.clear();
QUrl url(ftpServerLineEdit->text());
if (!url.isValid() || url.scheme().toLower() !=
  QLatin1String("ftp")) {
ftp->connectToHost(ftpServerLineEdit->text(), 21);
ftp->login();
} else {
ftp->connectToHost(url.host(), url.port(21));
if (!url.userName().isEmpty())
ftp-
  >login(QUrl::fromPercentEncoding(url.userName().toLa
  tin1()),
url.password());
else
ftp->login();
if (!url.path().isEmpty())
ftp->cd(url.path());
}
fileList->setEnabled(true);
connectButton->setEnabled(false);
connectButton->setText(tr("Disconnect"));
statusLabel->setText(tr("Connecting to FTP server
  %1...")
..arg(ftpServerLineEdit->text()));
}
```

كما ذكرنا هذه الدالة هي الأهم في موضوعنا كله في الأسطر من **2** الى **16** هو لقطع الإتصال في حال كان المتغير **ftp** مهيأ من قبل فهذا يدل على انه متصل وعلى اننا نريد ان نقفل هذا الإتصال فقمنا بإستدعاء الأمر **abort** الذي سوف يقوم بغلق أي أمر يشتغل وحذف جميع الأوامر المجدولة ومن ثم حذفنا موقعه في الذاكرة وأعطيناه القيمة صف لكي لايصبح مؤشرا هائما وقمنا بجعل جميع العناصر التي كانت ممكنة الى مقفلة لحين يعاد الإتصال ونضع النص **connect** للزر عوضا عن **disconnect** ومن ثم وضعنا صورة الفأرة الإفتراضية .

الأسطر من **17** الى **49** وهي في حالة لم يكن هناك إتصال ونريد ان نتصل بالخادم عن طريق بروتوكول **ftp** في بداية الأمر قمن بتغيير شكل مؤشر الفأرة الى حالة الإنتظار ثم هيأنا المتغير **ftp** ثم أصلناها بإشارات الإنتهاء والإبتداء بطلب والإتصال بصندوق شريط التحميل وعمليات الإتصال جميعها من الأسطر **21** الى **26** ثم قمنا بحذف كل ماتحتويه المتغيرات من معلومات قديمة في الأسطر من **27** الى **29** ثم صنعنا متغير من نوع **Qurl** ليحوي عنوان الخادم ثم وتأكدنا من العنوان ان كان صحيح وهل هو بروتوكول **ftp** ام لا ثم طلبنا الإتصال بالخادم في السطر **35** بإستخدام الدالة **connectToHost**

وطلبنا الدخول بإستخدام الدالة **login** في حال كان اسم المستخدم وكلمة المرور متوفرة مع عنوان الموقع يتم الدخول بإستخدامه في الأسطر من **36** الى **38** وغلا قام بمحاولة تسجيل الدخول بشكل طبيعي وتحويل المسار المرفق مع العنوان الى المسار الحلي بإستخدام الدالة **cd** في الأسطر من **39** الى **43** وفي نهاية الدالة قمنا بتحويل أي عنصر مقفل الى مفتوح وتغيير النص فوق زر **connect** الى **disconnect**.
ملاحظة : يمكنك إرفاق اسم المستخدم وكلمة المرور الى العنوان عن طريق **ftp://username:password@masfofa.com:port/path?vars**

دالة تحميل الملف

```
1. void FtpWindow::downloadFile()
2. {
3. QString fileName = fileList->currentItem()->text(0);
4. if (QFile::exists(fileName)) {
5. QMessageBox::information(this, tr("FTP"),
6. tr("There already exists a file called %1 in "
7. "the current directory.")
8. .arg(fileName));
9. return;
10.}
11.file = new QFile(fileName);
12.if (!file->open(QIODevice::WriteOnly)) {
13.QMessageBox::information(this, tr("FTP"),
14.tr("Unable to save the file %1: %2.")
15..arg(fileName).arg(file->errorString()));
16.delete file;
17.return;
18.}
19.ftp->get(fileList->currentItem()->text(0), file);
20.progressDialog->setLabelText(tr("Downloading
  %1...").arg(fileName));
21.downloadButton->setEnabled(false);
22.progressDialog->exec();
23.}
```

في البداية يتم تحديد اسم الملف الذي تريد تحميله عن طريق العنصر المختار من قائمة **fileList** في السطر الثالث.
ويمكننا بعدها ان نقول ان الدالة تنقسم لقسمين في حال كان الملف متوفر على جهازك من قبل فإنها ترسل رسالة بتوفر الملف وتخرج من الدالة وهي الأسطر من **4** الى **10**.
وفي حال لم يكن الملف موجود فإنه سوف يتم إكمال الدالة وتهيئة الدالة **file** وإعطائه المسار وفتحه للكتابة في حال فشلت عملية الفتح للكتابة يتم ارسال رسالة بفشل فتح الملف ويخرج من الدالة وهو في الأسطر من **12** الى **18** وفي حالة النجاح يتم تحميل الملف بواسطة إستخدام الدالة **get** ويتم الكتابة فوق شريط التحميل اسم الملف ويتم تنفيذ عملية التحميل بعد ذلك .

```
1. void FtpWindow::cancelDownload()
2. {
3. ftp->abort();
4. }
```

وهي عملية اقفال أي شغل تقوم به والمقصود به هنا هو التحميل.

```
1. void FtpWindow::ftpCommandFinished(int, bool error)
2. {
3. #ifndef QT_NO_CURSOR
4. setCursor(Qt::ArrowCursor);
5. #endif
6. if (ftp->currentCommand() == QFtp::ConnectToHost) {
7. if (error) {
8. QMessageBox::information(this, tr("FTP"),
9. tr("Unable to connect to the FTP server "
10."at %1. Please check that the host "
11."name is correct.")
12..arg(ftpServerLineEdit->text()));
13.connectOrDisconnect();
14.return;
15.}
16.statusLabel->setText(tr("Logged onto %1.")
17..arg(ftpServerLineEdit->text()));
18.fileList->setFocus();
19.downloadButton->setDefault(true);
20.connectButton->setEnabled(true);
21.return;
22.}
23.if (ftp->currentCommand() == QFtp::Login)
24.ftp->list();
25.if (ftp->currentCommand() == QFtp::Get) {
26.if (error) {
27.statusLabel->setText(tr("Canceled download of %1.")
28..arg(file->fileName()));
29.file->close();
30.file->remove();
31.} else {
32.statusLabel->setText(tr("Downloaded %1 to current
  directory.")
33..arg(file->fileName()));
34.file->close();
35.}
36.delete file;
37.enableDownloadButton();
38.progressDialog->hide();
39.} else if (ftp->currentCommand() == QFtp::List) {
40.if (isDirectory.isEmpty()) {
```

```
41.fileList->addTopLevelItem(new
   QTreeWidgetItem(QStringList() <<tr("<empty>")));
42.fileList->setEnabled(false);
43.}
44.}
45.}
```

هذه الدالة تستقبل اشارات نهاية أمر وتستقبل في وسائطها رقم الأمر ونتيجة العملية **true** اذا حدث خطأ ما **false** ان لم يحدث أي خطأ اثناء تنفيذ العملية وفي السطر الرابع ستجد انه تم تغيير حالة مؤشر الفأرة الى شكله الطبيعي لأن العملية أنتهت ثم استفسرنا عن العملية الحالية بداخل دالة الشرط ففي حالة كانت العملية الحالية هي الإتصال بالخادم أكملنا الشرط ثم قلنا اذا حدث هناك خطأ فسوف يتم ارسال رسالة الخطأ وتنفيذ الأمر **connectOrDisconnect** ويتم الخروج من الدالة وهذا جميعه في الأسطر من **6** الى **15** وفي الأسطر من **16** الى **22** يتم تمكين بعض العناصر وإعطاء بعضها نصوص وحديثنا هذا كله اذا كان الأمر هو الإتصال بالخادم
أي من الأسطر **6** الى **22** .
وبعد ذلك ان كانت العملية هي الدخول يتم تنفيذ الأمر **list** والذي سوف نتكلم يعيد عناوين الملفات في المسار الحالي الأسطر ما بين **23** الى **24.**
وإذا كان الأمر هو تحميل ملف واذا حدث خطأ يتم حذف الملف الذي تم صنعه لإستقباله ويتم ارسال رسالة الخطأ اما في حالة النجاح يتم وضع نص تبئ بأن عملية التحميل نجحت وهي في الأسطر من **25** الى **35** .
وبعد هذا كله يتم حذف عنون المؤشر **file** لإعطائه عنوان جديد فيما بعد لو اردت أن تحمل ملف جديد وبعده يتم إخفاء صندوق شريط التحميل وآخرا ان كان الأمر المنتهي هو **list** فإنه يتم التأكد من ان المسار فارغ او لا ان كان فارغ فمن الطبيعي يتم ارسال رسالة الفراغ وفي حالة كانت هناك ملفات سوف يتم أرسال اشارت **listinfo**() و في نهاية الأمر يتم إقفال قائمة المجلدات المتوفرة .

```
1. void FtpWindow::addToList(const QUrlInfo &urlInfo)
2. {
3. QTreeWidgetItem *item = new QTreeWidgetItem;
4. item->setText(0, urlInfo.name());
5. item->setText(1, QString::number(urlInfo.size()));
6. item->setText(2, urlInfo.owner());
7. item->setText(3, urlInfo.group());
8. item->setText(4,
   urlInfo.lastModified().toString("MMM dd yyyy"));
9. QPixmap pixmap(urlInfo.isDir() ?
   ":/images/dir.png" : ":/images/file.png");
10.item->setIcon(0, pixmap);
11.isDirectory[urlInfo.name()] = urlInfo.isDir();
12.fileList->addTopLevelItem(item);
13.if (!fileList->currentItem()) {
14.fileList->setCurrentItem(fileList->topLevelItem(0));
```

```
15.fileList->setEnabled(true);
16.}
17.}
```

ذكرنا في السابق ان الأمر **list** عند تنفيذه فإنه يتم ارسال اشارة
وهي **listInfo**() لكل ملف او مجلد وهذا الأمر هو من يستقبل هذه الإشارات
هذه الدالة تقوم بعمل بسيط جدا وهو تسجيل معلومات الملف داخل قائمة ملفات الموقع ووضع صورة ان كان ملف أو مجلد ويتم اضافته الى الأعلى القائمة وإن لم يكن عنصر مختار من القائمة يتم وضع الخيار على العنصر الأول وهو المضاف حديثا وفي الآخر يتم فتح القفل عنه .

```
1. void FtpWindow::processItem(QTreeWidgetItem *item,
   int /*column*/)
2. {
3. QString name = item->text(0);
4. if (isDirectory.value(name)) {
5. fileList->clear();
6. isDirectory.clear();
7. currentPath += "/" + name;
8. ftp->cd(name);
9. ftp->list();
10.cdToParentButton->setEnabled(true);
11.#ifndef QT_NO_CURSOR
12.setCursor(Qt::WaitCursor);
13.#endif
14.return;
15.}
16.}
```

في هذه الدالة قمنا بوضع اول عنوان من القائمة داخل المتغير **name** ثم قلنا ان توفرت كمسار داخل الحاوية فإننا نحذف جميع عناوين الحاوية والقائمة لنقوم بتعيينها كمسار حالي
وإستدعاء الأمر **list** ليعطينا جميع الملفات والمجلدات في المسار ووضعنا مؤشر الفأرة على شكل إنتظار .

```
1. void FtpWindow::cdToParent()
2. {
3. #ifndef QT_NO_CURSOR
4. setCursor(Qt::WaitCursor);
5. #endif
6. fileList->clear();
7. isDirectory.clear();
8. currentPath =
   currentPath.left(currentPath.lastIndexOf('/'));
9. if (currentPath.isEmpty()) {
10.cdToParentButton->setEnabled(false);
11.ftp->cd("/");
12.} else {
```

```
13.ftp->cd(currentPath);
14.}
15.ftp->list();
16.}
```

وهذه الدالة تستقبل اشارة زر العودة الى المجلد الأب
في البداية قمنا بتغيير مؤشر الفأرة الى الإنتظار ومن ثم حذفنا جميع محتوى قائمة الملف واحاوي **isDirectory** في الأسطر من **3** الى **7**
وفي ومن ثمنا أعطين للمتغير **currentPath** النص الكامل الأصلي له من اليسار الى آخر مكان وجد في الإشارة "/" فإذا كان فارغا جعلنا زر العودة الى الملف الأب مقفل ويتم تعيينه كمجلد حالي وإن كان المتغير به نص فإنه يعينه كمجلد حالي بدون اقفال زر او غيره ويستدعي الأمر **list** ليظهر ما بداخل المجلد .

```
1. void FtpWindow::updateDataTransferProgress(qint64
   readBytes,
2. qint64 totalBytes)
3. {
4. progressDialog->setMaximum(totalBytes);
5. progressDialog->setValue(readBytes);
6. }
```

وهي تستقبل إشارات من **dataTransferProgress**
وهذه الدالة تحدد اقصى قيمة لمربع شريط التحميل .
ومن ثم تعطيه قيمة في كل مرة يتم تحميل شيئ .

```
1. void FtpWindow::enableDownloadButton()
2. {
3. QTreeWidgetItem *current = fileList->currentItem();
4. if (current) {
5. QString currentFile = current->text(0);
6. downloadButton->setEnabled(!
   isDirectory.value(currentFile));
7. } else {
8. downloadButton->setEnabled(false);
9. }
10.}
```

وهي تحدد حالات يكون زر التحميل فيها مفتوح او مقفل وهنا اذا كان هناك عنصر تم إختياره فإنه يصبح مفتوح وقابل للضغط عليه وإلا تم إقفاله .

وبذلك تكون قد صنعت برنامج لنقل ملفات عبر بروتوكول نقل الملفات و شبه متكامل و في صف واحد فقط .

التعامل مع http (بروتوكول نقل النص التشعبي):

كما تحدثنا عن التعامل مع Ftp فإن http مثله ان لم يكن أسهل في التعامل منه لذلك لن تجد صعوبة في التعامل معه ان عرفت طريقة التعامل مع ftp ولكن كفكرة بسيطة عن http فإنه يقوم بنقل نصوص من ملف في الخادم الى ملف في جهازك ويكون ذلك عن طريق عناوين تحتوي على http:// او استخدام المنفذ 80 www.example.com:80 او إستخدام برنامج www .

وهذا البروتوكول يعمل بشكل غير متزامن فتجد أنه يرجع لك قبل ان يتم تنفيذه وينتظر دوره لكي يتم تنفيذه , ولذلك عليك ان تجعل البرنامج في حالة تكرار لكي يضمن لك معالجة طلبات httpالتي ارسلتها الى الخادم .

وهذا البروتوكول من حيث الدعم في Qt فإنه مغلفة ب QHttp ويمكنك ارسال طلباته عن طريق الصف الأخير مباشرة او عن طريق استخدام QhttpReqestHeader الذي بدوره يرث من QhttpHeader ويمكنك تلقي إجابة الطلب عن طريق الصف QhttpResponseHeader وهو أيضا يرث من QhttpHeaderوهناك صفوف أخرى قد تكون مفيدة بعض الأحيان مثل Qauthenticator وهو للتوثيق في العناوين والمنافذ التي قد تحتاج الى اسم مستخدم وكلمة مرور .

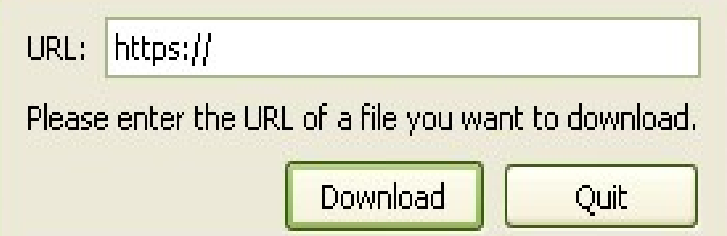
أهم الدوال المستخدمة:

setHoset تستطيع بهذه الدالة ان تحدد عنوان الخادم والمنفذ.

Get تقوم بجلب النصوص من العنوان .

Post تقوم بإرسال بيانات الى العنوان لتستقبل النصوص من العنوان .
Request لإرسال طلب عن طريق الصف QhttpReqestHeader.
setProxy لوضع وكيل وسيط بين طلبك وبين الخادم المقصود .
currentRequest يقابله currentCommand في ftp
read و readAll وهي تقوم بقراءة جميع النصوص المرسلة عن طريق الخادم
ويمكن استدعاءه عند صدور اشارة readyRead .

برنامج تحميل صفحات الإنترنت:



برنامج جدا بسيط في فكرته فهو يريد ان تعطيه عنوان الصفحة وبالمقابل يقوم هو بتحميله لك على جهازك ولكن دعنا نعرف كيف يعمل لكي تستطيع ان تعمل مثله ان اردت .

رأس الصف

```
1. #ifndef HTTPWINDOW_H
2. #define HTTPWINDOW_H
3. #include <QtGui>
4. #include <QtNetwork>
5. class HttpWindow : public QDialog
6. {
7. Q_OBJECT
8. public:
9. HttpWindow(QWidget *parent = 0);
10.private slots:
11.void downloadFile();
12.void cancelDownload();
13.void httpRequestFinished(int requestId, bool error);
14.void readResponseHeader(const QHttpResponseHeader
   &responseHeader);
```

```
15.void updateDataReadProgress(int bytesRead, int
   totalBytes);
16.void enableDownloadButton();
17.void slotAuthenticationRequired(const QString &,
   quint16, QAuthenticator *);
18.#ifndef QT_NO_OPENSSL
19.void sslErrors(const QList<QSslError> &errors);
20.#endif
21.private:
22.QLabel *statusLabel;
23.QLabel *urlLabel;
24.QLineEdit *urlLineEdit;
25.QProgressDialog *progressDialog;
26.QPushButton *downloadButton;
27.QPushButton *quitButton;
28.QDialogButtonBox *buttonBox;
29.QHttp *http;
30.QFile *file;
31.int httpGetId;
32.bool httpRequestAborted;
33.};
34.#endif
```

غير دالة البناء فإن جميع الدوال هنا هي مستقبلات لإشارات مثل تحميل و إلغاء التحميل و عند إنتهاء طلب و قراء الإجابة على طلب و شريط التحميل وو...
ومتغيرات أهمها هو QHttp فتتبعها.

جسم الصف

```
1. HttpWindow::HttpWindow(QWidget *parent)
2. : QDialog(parent)
3. {
4. #ifndef QT_NO_OPENSSL
5. urlLineEdit = new QLineEdit("https://");
6. #else
7. urlLineEdit = new QLineEdit("http://");
8. #endif
9. urlLabel = new QLabel(tr("&URL:"));
10.urlLabel->setBuddy(urlLineEdit);
11.statusLabel = new QLabel(tr("Please enter the URL of a file
   you want to "
12."download."));
```

```
13.downloadButton = new QPushButton(tr("Download"));
14.downloadButton->setDefault(true);
15.quitButton = new QPushButton(tr("Quit"));
16.quitButton->setAutoDefault(false);
17.buttonBox = new QDialogButtonBox;
18.buttonBox->addButton(downloadButton,
   QDialogButtonBox::ActionRole);
19.buttonBox->addButton(quitButton,
   QDialogButtonBox::RejectRole);
20.progressDialog = new QProgressDialog(this);
21.http = new QHttp(this);
22.connect(urlLineEdit, SIGNAL(textChanged(const QString &)),
23.this, SLOT(enableDownloadButton()));
24.connect(http, SIGNAL(requestFinished(int, bool)),
25.this, SLOT(httpRequestFinished(int, bool)));
26.connect(http, SIGNAL(dataReadProgress(int, int)),
27.this, SLOT(updateDataReadProgress(int, int)));
28.connect(http, SIGNAL(responseHeaderReceived(const
   QHttpResponseHeader &)),
29.this, SLOT(readResponseHeader(const QHttpResponseHeader &)));
30.connect(http, SIGNAL(authenticationRequired(const QString &,
   quint16, QAuthenticator *)),
31.this, SLOT(slotAuthenticationRequired(const QString &,
   quint16, QAuthenticator *)));
32.#ifndef QT_NO_OPENSSL
33.connect(http, SIGNAL(sslErrors(const QList<QSslError> &)),
34.this, SLOT(sslErrors(const QList<QSslError> &)));
35.#endif
36.connect(progressDialog, SIGNAL(canceled()), this,
   SLOT(cancelDownload()));
37.connect(downloadButton, SIGNAL(clicked()), this,
   SLOT(downloadFile()));
38.connect(quitButton, SIGNAL(clicked()), this, SLOT(close()));
39.QHBoxLayout *topLayout = new QHBoxLayout;
40.topLayout->addWidget(urlLabel);
41.topLayout->addWidget(urlLineEdit);
42.QVBoxLayout *mainLayout = new QVBoxLayout;
43.mainLayout->addLayout(topLayout);
44.mainLayout->addWidget(statusLabel);
45.mainLayout->addWidget(buttonBox);
46.setLayout(mainLayout);
47.setWindowTitle(tr("HTTP"));
48.urlLineEdit->setFocus();
49.}
```

10-1في بداية الأمر تجد أننا هيأنا مربع العنوان ان كان https:// او http:// وبعدها هيأنا label تعريف مربع العنوان وقمنا بربطه مع موضع العنوان.

11-21قمنا بتهيئة label الحالة وبتهيئة زر التحميل وجعلناه في حالة تلقي حدث المفتاح enter وبعدها تم تهيئة زر الخروج من البرنامج وصندوق الأزرار ليضاف اليه الزرين التحميل والإقفال وهيأنا بعد ذلك صندوق شريط التحميل وآخر ما تم تهيئته هو المتغير الأهم وهو QHttp .
22-38 قمن بصنع عمليات ربط بين الإشارات والمستقبلات وهذا ما سوف تتعرف عليه خلال رحلتنا داخل البرنامج .
38-49 قمنا بترتيب العناصر بإستخدام layout لنقوم بعد ذلك بإعطاء عنوان للنافذة ووضع التركيز على مربع العنون .

دالة طلب التحميل

```
1. void HttpWindow::downloadFile()
2. {
3. QUrl url(urlLineEdit->text());
4. QFileInfo fileInfo(url.path());
5. QString fileName = fileInfo.fileName();
6. if (fileName.isEmpty())
7. fileName = "index.html";
8. if (QFile::exists(fileName)) {
9. if (QMessageBox::question(this, tr("HTTP"),
10.tr("There already exists a file called %1 in "
11."the current directory. Overwrite?").arg(fileName),
12.QMessageBox::Ok|QMessageBox::Cancel,
   QMessageBox::Cancel)
13.== QMessageBox::Cancel)
14.return;
15.QFile::remove(fileName);
16.}
17.file = new QFile(fileName);
18.if (!file->open(QIODevice::WriteOnly)) {
19.QMessageBox::information(this, tr("HTTP"),
20.tr("Unable to save the file %1: %2.")
21..arg(fileName).arg(file->errorString()));
22.delete file;
23.file = 0;
24.return;
25.}
26.QHttp::ConnectionMode mode = url.scheme().toLower()
   == "https" ? QHttp::ConnectionModeHttps :
   QHttp::ConnectionModeHttp;
27.http->setHost(url.host(), mode, url.port() == -1 ? 0
   : url.port());
28.if (!url.userName().isEmpty())
29.http->setUser(url.userName(), url.password());
30.httpRequestAborted = false;
31.QByteArray path =
   QUrl::toPercentEncoding(url.path(), "!
   $&'()*+,;=:@/");
```

```
32.if (path.isEmpty())
33.path = "/";
34.httpGetId = http->get(path, file);
35.progressDialog->setWindowTitle(tr("HTTP"));
36.progressDialog->setLabelText(tr("Downloading
  %1.").arg(fileName));
37.downloadButton->setEnabled(false);
38.}
```

هذه الدالة تستقبل إشارات من زر التحميل.
3-16قمنا بادئ الأمر بإستقبال العنوان من مربع العنوان في المتغير Qurl ومن ثم تم اضافة المسار الى متغير من نوع QfileInfo ليعطينا بعض المعلومات بعد ذلك وبعدها قمنا بصنع المتغير fileName ليأخذ اسم الملف الذي نريد تحميله ونختبره إن كان فارغا فهذا يدل على ان اسمه هو index.html ثم نختبر هل الملف موجود من قبل على جهازنا ؟؟ ان كان موجود يتم إرسال رسالة عن وجوده وهل تريد إعادة تحميله من جديد وحذف الملف القديم او الغاء الطلب ؟ في حالة طلب الإلغاء يخرج من الأمر مباشرة وإلا فإنها يقوم بحذف الملف السابق ليتم عملية التحميل من جديد فيما بعد.
17-25 يتم صنع ملف جديد ليستقبل ما يتم تحميله وفتحه للكتابة وإن فشل سوف يتم ارسال رسالة الفشل وحذف عنوان مؤشر المتغير file ويتم الخروج من الدالة .
25-30وإلا نبدأ في عملية التحميل في بداية الأمر يجب أن نحدد نوع الإتصال بالبروتوكول هل هو http الإفتراضي ام أنه في ftps الأكثر أمانا ؟ وهذا ماسخرت له الأسطر التالية حيث يتم تحديد نوع بروتوكول ftp عن طريق التدقيق على العنوان وهل هناك اسم مستخدم او كلمة مرور لكي يتم بعد ذلك إضافتها .
31-38 يتم اعادة ترميز المسار لتنقيحه الى ascii من الرموز الغير معترف بها في المسار والمعرفة في المحارف التالية " "!$&'()*+,:=;@/ثم تم إختباره ان كان فارغا فإن تحقق الشرط يتم تحوير المسار الى عنوان الموقع وهذا يعني الرجوع الى index.html وأخيرا قمنا بعملية تحميل الملف وتغيير النصزص فوق شريط التحميل وجعل زر التحميل مقفل .

دالة طلب الإلغاء

```
void HttpWindow::cancelDownload()
 {
     statusLabel->setText(tr("Download canceled."));
     httpRequestAborted = true;
     http->abort();
     downloadButton->setEnabled(true);
 }
```

وهي تستقبل مؤشرات طلب الإلغاء .
ويتم فيها الخروج من العملية الحالية وحذف جميع الطلبات المجدولة وتمكين زر التحميل مرة أخرى .

دالة الإنتهاء من طلب

```
1. void HttpWindow::httpRequestFinished(int requestId,
   bool error)
2. {
3. if (requestId != httpGetId)
4. return;
5. if (httpRequestAborted) {
6. if (file) {
7. file->close();
8. file->remove();
9. delete file;
10.file = 0;
11.}
12.progressDialog->hide();
13.return;
14.}
15.if (requestId != httpGetId)
16.return;
17.progressDialog->hide();
18.file->close();
19.if (error) {
20.file->remove();
21.QMessageBox::information(this, tr("HTTP"),
22.tr("Download failed: %1.").arg(http-
   >errorString()));
23.} else {
24.QString fileName = QFileInfo(QUrl(urlLineEdit-
   >text()).path()).fileName();
25.statusLabel->setText(tr("Downloaded %1
   tocurrentdirectory. ").arg(fileName));
26.}
27.downloadButton->setEnabled(true);
28.delete file;
29.file = 0;
30.}
```

وهي تستقبل إشارات الإنتهاء من طلب
1-18 أولى الخطوات للدالة هي التأكد هل كانت العملية المنتهية هي تحميل ملف أم لا وذلك بإستخدام رقم العملية؟؟ ان لم تكن سوف يتم الخروج من الدالة وإلا يتم التأكد هلى كان السبب هو طلب الإلغاء أم لا ؟؟
ان كان كذلك يتم حذف الملف وحذف عنوان المؤشر وإخفاء صندوق شريط التحميل والخروج من الدالة وإلا

سوف يم تم إخفاء شريط التحميل وإقفال الملف بعد الكتابة عليه .
15-30 في حال حدوث خطا اثناء التحميل يتم حذف الملف وإرسال رسالة الخطأ وإلا يتم تعيين اسم الملف وإرسال رسالة تخبرك بأنه تمت العملية وجعل زر التحميل ممكنا وحذف عنوان المؤشر file .

```
1. void HttpWindow::readResponseHeader(const
   QHttpResponseHeader &responseHeader)
2. {
3. switch (responseHeader.statusCode()) {
4. case 200:                   // Ok
5. case 301:                   // Moved Permanently
6. case 302:                   // Found
7. case 303:                   // See Other
8. case 307:                   // Temporary Redirect
9. // these are not error conditions
10.break;
11.default:
12.QMessageBox::information(this, tr("HTTP"),
13.tr("Download failed:
   %1.").arg(responseHeader.reasonPhrase()));
14.httpRequestAborted = true;
15.progressDialog->hide();
16.http->abort();
17.}
18.}
```

وتستقبل اشاراة المستقبلة من الخادم .
في حال كان احد الأرقام التي تم تعيينها فهذا يعني أن طلب التحميل ليس عليه ضرر وإلا فإن التحميل فشل ويتم الخروج بقوة من العملية وإخفاء شريط التحميل.

دالة تقدم شريط التحميل

```
1. void HttpWindow::updateDataReadProgress(int
   bytesRead, int totalBytes)
2. {
3. if (httpRequestAborted)
4. return;
5. progressDialog->setMaximum(totalBytes);
6. progressDialog->setValue(bytesRead);
7. }
```

ويستقبل إشارات تحدد كمية البيانات التي تم تحميلها فهو بدوره يترجمها الى شريط التحميل .

دالة تحدد متى يمكن او يقفل زر التحميل

```
1. void HttpWindow::enableDownloadButton()
2. {
3. downloadButton->setEnabled(!urlLineEdit-
   >text().isEmpty());
4. }
```

في حال كان مربع العنوان فارغ يتم إقفال زر التحميل والعكس صحيح.

```
1. void HttpWindow::slotAuthenticationRequired(const
   QString &hostName, quint16, QAuthenticator
   *authenticator)
2. {
3. QDialog dlg;
4. Ui::Dialog ui;
5. ui.setupUi(&dlg);
6. dlg.adjustSize();
7. ui.siteDescription->setText(tr("%1 at
   %2").arg(authenticator->realm()).arg(hostName));
8. if (dlg.exec() == QDialog::Accepted) {
9. authenticator->setUser(ui.userEdit->text());
10.authenticator->setPassword(ui.passwordEdit->text());
11.}
12.}
```

وتستقبل إشارة لوجوب توفر اسم مستخدم وكلمة مرور لكي تنفذ اليها .

في هذه الدالة قاموا قبل ذلك بصنع النافذة بواسطة الأداة Qt Designer في حال كان من الضروري إضافة اسم مستخدم وكلمة مرور للنفاذ الى الموقع فإن هذه النافذة يتم فتحها .

وهذه الخدمة يقدمها الصف QAuthenticator .

```
1. #ifndef QT_NO_OPENSSL
2. void HttpWindow::sslErrors(const QList<QSslError>
   &errors)
3. {
4. QString errorString;
5. foreach (const QSslError &error, errors) {
6. if (!errorString.isEmpty())
7. errorString += ", ";
8. errorString += error.errorString();
9. }
```

```
10.if (QMessageBox::warning(this, tr("HTTP Example"),
11.tr("One or more SSL errors has occurred:
   %1").arg(errorString),
12.QMessageBox::Ignore | QMessageBox::Abort) ==
   QMessageBox::Ignore) {
13.http->ignoreSslErrors();
14.}
15.}
16.#endif
```

وتستقبل اشارات أخطاء SSL
في بداية الأمر تم تحويل جميع الأخطاء الى نصوص تحتويها المتغير errorString وترسل رسالة ما ان أراد ان يتجاهل الخطأ ويكمل العملية وإلا انه يتوقف.

اضافة :
يمكنك وضع الطلبات بإستخدام الصف QhttpResponseHeader عوضا عن الدوال المتوفر في http انظر هذا المثال :

```
1. QHttpRequestHeader header("POST", "/search.html");
2. header.setValue("Host", "www.trolltech.com");
3. header.setContentType("application/x-www-form-
   urlencoded");
4. http.setHost("www.trolltech.com");
5. http.request(header, "qt-
   interest=on&search=opengl");
```

لقد أصبح بمقدورك الآن ان تصنع برنامج يستخدم بروتوكول http بإحتراف.

Sockets

وكتعريف عام لها يمكنك قول ان ال socket هي قناة اتصال منطقية تفتح بين طرفين بهدف تبادل المعطيات .
وهو يعتمد في إتصاله هذا على عنوان و منفذ وتتعامل مع بروتوكولات عدة مثل udp و tcp و ssl ولو تمعنت النظر في البروتوكولات السابقة تجد انها عبارة عن socket
ولكن ركبت بطريقة تؤدي الغرض المطلوب منها وكنظرة عامة فإن أي تناقل بيانات في الشبكة في الغالب هو عبارة عن socket مالذي يعنيه هذا لك؟؟
في الحقيقة لو أنك قرأت هذا القسم فقط من الباب بأكمله فأقل لك أحسنت الإختيار لأن هذا يعني أنه يمكن ارسال وإستقبال البريد الإلكتروني او عن طريق النقاش الحي بإستخدام عناوين ومنافذ البريد الإلكتروني وتبادل المقاطع المرئية عبر الشبكة أو حتى صنع لعبة بين مجموع من الأفراد على شبكة واحدة لها نفس العنوان والمنفذ ليس هذا فقط !!! بل كل مايخطر ببالك من تفاعل عبر الشبكة .

مالفرق بين إستخدامها مع udp او tcp ؟

تخيل لو أنك قمت بجلب بيانات صورة ما عبر الشبكة بإستخدام socket مرة مع بروتوكول tcp ومرة أخرى مع بروتوكول udp مالفرق ؟؟
ستجد أن الصورة التي جلبتها بواسطة بروتوكول tcp مطابقة للصورة الأصلية بالرغم من تأخرها في إرسال البيانات اليك ولكن في المقابل تجد ان udp ارسلت لك البيانات في غمضة عين ولكن !!
ستجد أن هناك فراغات سوداء في الصورة والسبب واضح أن الأول مصيره ان يصل لك إن شاء الله اما الثاني فهو يرسل البيانات الى عنوانك ولا تدري أسوف تصل كاملة ام يذهب بعضها الى المجهول اثناء رحلة السفر عبر الشبكة , وهذا ما ميز البروتوكول tcp عنها حيث لو صار عطب في أي من الأجزاء يتم تعويضه الى ان تصل بياناتك كاملة سليمة .

الخادم والعميل :

وفرت لك Qt في أن تتعامل مع جميع هذه الأنواع من الإتصالات لكي تأخذ الحرية الكاملة في إنشاء الشبكة التي تريدها فقدمت لك صفوف تدعم الخادم وصفوف أخرى تدعم العميل وأيضا شبكات الند للند فمثلا إن اردت أن تجعل مجموعة من الأجهزة تعمل على طابعة واحدة فأن في هذه الحالة بحاجة الى خادم والخادم يكون محلي لأنها شبكة محلية و.... و....وو في الحقيقة لا داعي التفكير المطول في هذه القضية فقط وفرت Qt لك الصف QtcpServer حيث يوفر لك عنوان وهو localhost و منفذ ويأتي متغيرا ويكتشف لك البرامج أو جهات الإتصال التي ولجت إليه و الجهات المسموحة لها بالدخول أو لا وو كل هذا في صف وحيد بما أنه يرسل لك عنونا و منفذ فبإمكانك صنع برامج أو توصيلها عبر هذا المنفذ فهو يرسل الى الخادم ويستقبل منها وبالطبع هذا في صف واحد أيضا وهو QabstractSocket والذي يرث من QIODevice ويرث من الأول كلا من QtcpSocket للتعامل مع بروتوكول tcp و QudpSocket للتعامل مع بروتوكول udp وهكذا يكون قد أخذت نظرة عامة على ماقد تستطيع فعله إذا ماتعلمت Qt بإذن الله.

التعامل مع tcp (بروتوكول التحكم بالإرسال):

برنامج اللإرسال والإستقبال tcp :

هو في الحقيقة ليس برنامج إنما برنامجين الأول للخادم والثاني للعميل وكلما حدث إتصال جديد مع الخادم يقوم

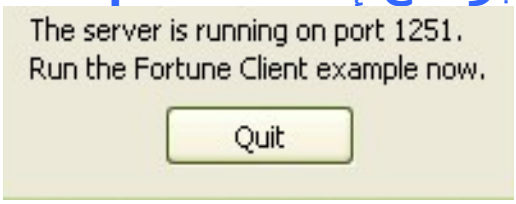
بإرسال نص ليستقبله العميل وهذه الأنواع من البرامج هي الأفضل لتوصيل الفكرة للقارئ.

الخادم QTcpServer:

أهم الدوال:

Listen لإضافة عنوان ومنفذ الى الخادم في حال عدم اضافة شيئ له يتم استقبال أي منفذ وأي عنوان لخادم .
isListen في حال انه يستمع للإتصالات القادمة يعيد القيمة true .
serverPort ويرجع قيمة المنفذ للخادم .
nextPendingConnection وترجع عميل متصل مع الخادم ويتم حذفه عند حذف الخادم .
newConnection وهي إشارة يتم إرسالها في كل مرة يتصل عميل جيد بالخادم .
setMaxPendingConnection ويحدد أعلى طاقة يستوعبها الخادم من العملاء وهو في حالته الإفتراضية 30 .

برنامج إنشاء الخادم



من الصورة تستنتج ان منفذ الخادم هو 1251 وهو ليس ثابت فلو قمت بإغلاق البرنامج وفتحته مرة أخرى فإنك سوف تجد ان المنفذ تغير .
تستطيع ان تصنع الخادم في سطرين برمجيين بواسطة Qt ولكن إن أردت خادم يكتشف الإتصالات ويتعامل معها في هذه الحالة لا بأس ان زدت الشفرة المصدرية قليلا.

رأس الصف

```
1. #ifndef SERVER_H
2. #define SERVER_H
3. #include <QtWidget>
4. #include <QtNetwork>
5. class Server : public QDialog
6. {
7. Q_OBJECT
8. public:
9. Server(QWidget *parent = 0);
10.private slots:
11.void sendFortune();
12.private:
13.QLabel *statusLabel;
```

```
14.QPushButton *quitButton;
15.QTcpServer *tcpServer;
16.QStringList fortunes;
17.};
18.#endif
```

صنعنا رأس الصف وهو يحتوي بجانب دالة البناء على دالة أخرى من نوع مستقبل إشارات والتي ستكتشف فيا بعد ان هذه الإشارة هي متصل جديد ويوجد أيضا متغيرات ومؤشرات أهمها tcpServer فضع عينك عليه .

جسم الملف :

```
1. Server::Server(QWidget *parent)
2. : QDialog(parent)
3. {
4. statusLabel = new QLabel;
5. quitButton = new QPushButton(tr("Quit"));
6. quitButton->setAutoDefault(false);
7. tcpServer = new QTcpServer(this);
8. if (!tcpServer->listen()) {
9. QMessageBox::critical(this, tr("Fortune Server"),
10.tr("Unable to start the server: %1.")
11..arg(tcpServer->errorString()));
12.close();
13.return;
14.}
15.statusLabel->setText(tr("The server is running on
   port %1.\n"
16."Run the Fortune Client example now.")
17..arg(tcpServer->serverPort()));
18.fortunes << tr("You've been leading a dog's life.
   Stay off the furniture.")
19.<< tr("You've got to think about tomorrow.")
20.<< tr("You will be surprised by a loud noise.")
21.<< tr("You will feel hungry again in another hour.")
22.<< tr("You might have mail.")
23.<< tr("You cannot kill time without injuring
   eternity.")
24.<< tr("Computers are not intelligent. They only
   think they are.");
25.connect(quitButton, SIGNAL(clicked()), this,
   SLOT(close()));
26.connect(tcpServer, SIGNAL(newConnection()), this,
   SLOT(sendFortune()));
27.QHBoxLayout *buttonLayout = new QHBoxLayout;
28.buttonLayout->addStretch(1);
29.buttonLayout->addWidget(quitButton);
30.buttonLayout->addStretch(1);
31.QVBoxLayout *mainLayout = new QVBoxLayout;
```

```
32.mainLayout->addWidget(statusLabel);
33.mainLayout->addLayout(buttonLayout);
34.setLayout(mainLayout);
35.setWindowTitle(tr("Fortune Server"));
36.}
```

4-14 في بادئ الأمر قمنا بتهيئة المتغيرات وجعل زر الخروج لا يتلقى المفتاح enter
ثم قمنا بتهيئة المتغير tcpServer لنستخدم بعدها الدالة listen في حال تم النجاح في عملية صنع الخادم فإنه يعيد القية true وينشئ بعدها منفذ عشوائي لتستخدمه بعد ذلك العميل ليتصل بالخادم ولكن هنا أختبرنا ما ان وقع الفشل في إنشاء الخادم فإنه يتم إرسال رسالة بالفشل والخروج من البرنامج .
15-24 أما في حالة نجاح إنشاء الخادم يتم كتابة النص على النافذة انه تم الإتصال وهذا رقم المنفذ وثم قمنا بوضع النصوص التي سوف نرسلها الى العميل داخل حاوية نصوص .
25-36 قمنا فيها بعمليات الربط بين الإشارات والمستقبلات والإشارات هي ضغط زر الخروج وإتصال عميل جديد بالخادم ثم قمنا بترتيب العناصر داخل النافذة الرئيسية بإستخدام layout وفي الآخر قمنا بإعطاء عنوان للنافذة .

```
1. void Server::sendFortune()
2. {
3. QByteArray block;
4. QDataStream out(&block, QIODevice::WriteOnly);
5. out.setVersion(QDataStream::Qt_4_0);
6. out << (quint16)0;
7. out << fortunes.at(qrand() % fortunes.size());
8. out.device()->seek(0);
9. out << (quint16)(block.size() - sizeof(quint16));
10.QTcpSocket *clientConnection = tcpServer-
   >nextPendingConnection();
11.connect(clientConnection, SIGNAL(disconnected()),
12.clientConnection, SLOT(deleteLater()));
13.clientConnection->write(block);
14.clientConnection->disconnectFromHost();
15.}
```

9-3كما ذكرنا مسبقا ان هذه الدالة تلتقط إشارات متصل جديد الى الخادم قمنا في بادئ الأمر بصنع متغير للتيارات الثنائية QDataStream وأعطيناه متغير من النوع QbyteArray ليكتب فقط ثم وضعنا إصدار Qt .

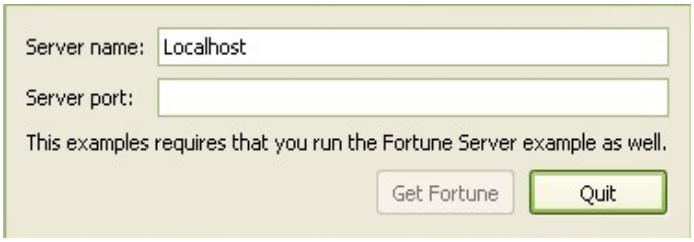
وكتبنا لها الرقم صفر على أنه 2 بايت وأدخلنا لها مقولة عشوائية من المقولات التي صنعناها في الحاوية وقمنا بعد ذلك بكتابة حجم النص المنقول .
10-15 قمنا بإستدعاء العميل المتوفر للخادم بال socket لكي نتمكن من الكتابة والقراءة في الخادم ومنه.
وطلبنا في السطر 12 بحذف الصف لاحقا أي بعد اول عملية تكرار للبرنامج ثم طلبنا كتابة النص الى الخادم لنتمكن من قراءته بعد ذلك عبر برنامج العميل لاحقا وقطعنا اتصال ال socket مع الخادم .
ملاحظة: في البرنامج أستعنا بالمتصلين بالخادم لكي نكتب النص إليه من الأسطر 10-14 .

برنامج العميل بالإعتماد على بروتوكول tcp QtcpSocket,QabstractSocket:

أهم الدوال:

الصف QtcpSocket يرث من الصف QabstractSocket وهو بدوره يرث من QIODevice.
connectToHost الإتصال بالخادم بإستخدام العنوان والمنفذ.
disconnectFromHost قطع الإتصال مع الخادم .
connected اشارة يتم ارسالها في حالة الإتصال بالخادم.
Disconnected اشار يتم ارسالها في حالة قطع الإتصال بالخادم.
bytesAvailabel وتعيد عدد البيتات المتوفرة للقراءة .
error اشارات يتم إرسالها في حال وجود أخطاء .
وهناك دوال أخرى ترثها من QIODevice :
readyRead اشارة يتم ارسالها في حال توفر بيانات لقراءتها .
Read و readAll للقراءة.
Write للكتابة الى الخادم .

برنامج العميل:



في برنامج الخادم قمنا بإنشاء الخادم بعنوان ومنفذ عشوائي وتحدثنا ان فكرة البرنامج هو إرسال نصوص الى الخادم وقراءتها بواسطة العميل ولكن في برنامجين.
فالوظيفة المطلوبة من هذا البرنامج هو قراءة النصوص من الخادم وبعدها يقوم بقطع الإتصال ليتصل مرة أخرى عند طلب نص آخر ويكتشف الخادم إتصال جديد لتيدأ عملية ارسال نص جديد للخادم ومن ثم يقرأها العميل بعد ذلك كملخص عند كل عملية قطع اتصال والإتصال مرة أخرى .

رأس الصف:

```
1. #ifndef CLIENT_H
2. #define CLIENT_H
3. #include <QtGui>
4. #include <QtNetwork>
5. class Client : public QDialog
6. {
7. Q_OBJECT
8. public:
9. Client(QWidget *parent = 0);
10.private slots:
11.void requestNewFortune();
12.void readFortune();
13.void displayError(QAbstractSocket::SocketError
   socketError);
14.void enableGetFortuneButton();
15.private:
16.QLabel *hostLabel;
17.QLabel *portLabel;
18.QLineEdit *hostLineEdit;
19.QLineEdit *portLineEdit;
20.QLabel *statusLabel;
```

```
21.QPushButton *getFortuneButton;
22.QPushButton *quitButton;
23.QDialogButtonBox *buttonBox;
24.QTcpSocket *tcpSocket;
25.QString currentFortune;
26.quint16 blockSize;
27.};
28.#endif
```

غير دالة البناء فإن جميع الدوال الأخرى هي عبارة عن مستقبلات لإشارات وقد لاحظنا هذا في أمثلتنا كلها وذلك لعدم توفر الأحداث الكافية في هذه الصفوف ثم صنعنا متغيراتنا التي نحتاجها وأهمها بالطبع tcpSocket فضع عينيك عليها.

جسم الصف :

```
1. Client::Client(QWidget *parent)
2. : QDialog(parent)
3. {
4. hostLabel = new QLabel(tr("&Server name:"));
5. portLabel = new QLabel(tr("S&erver port:"));
6. hostLineEdit = new QLineEdit("Localhost");
7. portLineEdit = new QLineEdit;
8. portLineEdit->setValidator(new QIntValidator(1,
   65535, this));
9. hostLabel->setBuddy(hostLineEdit);
10.portLabel->setBuddy(portLineEdit);
11.statusLabel = new QLabel(tr("This examples requires
   that you run the "
12."Fortune Server example as well."));
13.getFortuneButton = new QPushButton(tr("Get
   Fortune"));
14.getFortuneButton->setDefault(true);
```

```
15.getFortuneButton->setEnabled(false);
16.quitButton = new QPushButton(tr("Quit"));
17.buttonBox = new QDialogButtonBox;
18.buttonBox->addButton(getFortuneButton,
   QDialogButtonBox::ActionRole);
19.buttonBox->addButton(quitButton,
   QDialogButtonBox::RejectRole);
20.tcpSocket = new QTcpSocket(this);
21.connect(hostLineEdit, SIGNAL(textChanged(const
   QString &)),
22.this, SLOT(enableGetFortuneButton()));
23.connect(portLineEdit, SIGNAL(textChanged(const
   QString &)),
24.this, SLOT(enableGetFortuneButton()));
25.connect(getFortuneButton, SIGNAL(clicked()),
26.this, SLOT(requestNewFortune()));
27.connect(quitButton, SIGNAL(clicked()), this,
   SLOT(close()));
28.connect(tcpSocket, SIGNAL(readyRead()), this,
   SLOT(readFortune()));
29.connect(tcpSocket,
   SIGNAL(error(QAbstractSocket::SocketError)),
30.this,
   SLOT(displayError(QAbstractSocket::SocketError)));
31.QGridLayout *mainLayout = new QGridLayout;
32.mainLayout->addWidget(hostLabel, 0, 0);
33.mainLayout->addWidget(hostLineEdit, 0, 1);
34.mainLayout->addWidget(portLabel, 1, 0);
35.mainLayout->addWidget(portLineEdit, 1, 1);
36.mainLayout->addWidget(statusLabel, 2, 0, 1, 2);
37.mainLayout->addWidget(buttonBox, 3, 0, 1, 2);
38.setLayout(mainLayout);
39.setWindowTitle(tr("Fortune Client"));
40.portLineEdit->setFocus();
41.}
```

4-7 قمنا بتهيئة المتغيرات .
8-15في موضع المنفذ في البرنامج قمنا بوضع ضمانات رقمية لن يقل الرقم عن 1 ولن يزيد عن 65535 وربطناهم بعد ذلك بال label معرف لهم حتى اذا ماتم الضغط على الزر alt+الحرف الذي تحته خط يتم الإنتقال الى مربع النص المرتبط بها وجعلنا زر التقاط النص في حالة إستقبال للمفتاح enter, وجعلناه ايضا في حالة مقفلة.
16-25 هيأنا زر وأنشأنا صدوق أزرار لينظم الاول معهم ومن ثم قمنا بتهيئة المتغير الأهم وهو tcpsocket وبعد ذلك قمنا بعمليات ربط بين الإشارات والمستقبلا ثم نظمنا النافذة الرئيسية بإستخدام layout ووضعنا التركيز على مربع العنوان .

```
1. void Client::requestNewFortune()
2. {
3. getFortuneButton->setEnabled(false);
4. blockSize = 0;
5. tcpSocket->abort();
6. tcpSocket->connectToHost(hostLineEdit->text(),
7. portLineEdit->text().toInt());
8. }
```

تستقبل إشارات زر طلب النص
في البداية قمنا بجعل الزر في حالة قفل حتى ننتهي من المعالجة و قمنا بالخروج من الإتصال الحاليلنقوم بالإتصال مرة أخرى بالخادم على العنوان والمنفذ ذاته وكما عرفنا من برنامج الخادم ان كل عملية إتصال جديدة ينتج عنها توليد نص .

```
1. void Client::readFortune()
2. {
3. QDataStream in(tcpSocket);
4. in.setVersion(QDataStream::Qt_4_0);
5. if (blockSize == 0) {
6. if (tcpSocket->bytesAvailable() <
   (int)sizeof(quint16))
7. return;
8. in >> blockSize;
9. }
10.if (tcpSocket->bytesAvailable() < blockSize)
11.return;
12.QString nextFortune;
13.in >> nextFortune;
14.if (nextFortune == currentFortune) {
15.QTimer::singleShot(0, this,
   SLOT(requestNewFortune()));
16.return;
17.}
18.currentFortune = nextFortune;
19.statusLabel->setText(currentFortune);
20.getFortuneButton->setEnabled(true);
21.}
```

9-3 بعد ان أصبح في الخادم نصوص جديدة فإنه تم إرسال اشارات لهذا المستقبل عن توفر نصوص جديدة فصنعنا متغير تيارات ثنائية ليقرأ من ال **socket** ووضعنا إصدار ال **Qt** وقلنا ان كان المتغير **blockSize** فارغ وإذا كان حجم النص المتوفر للقراءة أصغر من **2** بايت فإنه يتم الخروج من الدالة وإلا تم قراءة الرقم والذي هو في الحقية يمثل حجم النص الذي تم تحويله .
10-17 بدأنا شرط جديد وهو هل البيتات المتوفرة للقراءة أقل من البيتات المعلن عنها إذا كان كذلك يتم الخروج من الدالة

وإلا فإنه يكمل ثم قمنا بإنشاء متغير نص وقرأنا النص من الخادم بأكمله ثم أختبرنا هل النص الذي قمنا بسحبه مشابه للنص الحالي المكتوب في النافذة ان كانت الإجابة نعم فإنه يجب ان نوفر نص آخر فقمنا بإرسال مؤقت على الكائن الرئيسي ليقوم بقذف إشارة الى طلب نص جديد وثم الخروج.
في هذه الحالة سوف يكون هناك إشارة توفر نص جديد للقراءة تم قذفها ليتم لنشهد دالة جديدة مثلها .
18-21وإلا تم كتابة النص على النافذة الرئيسية وإرجاع زر طلب نص جديد في حالة تمكين .

```
1. void
   Client::displayError(QAbstractSocket::SocketError
   socketError)
2. {
3. switch (socketError) {
4. case QAbstractSocket::RemoteHostClosedError:
5. break;
6. case QAbstractSocket::HostNotFoundError:
7. QMessageBox::information(this, tr("Fortune Client"),
8. tr("The host was not found. Please check the "
9. "host name and port settings."));
10.break;
11.case QAbstractSocket::ConnectionRefusedError:
12.QMessageBox::information(this, tr("Fortune Client"),
13.tr("The connection was refused by the peer. "
14."Make sure the fortune server is running, "
15."and check that the host name and port "
16."settings are correct."));
17.break;
18.default:
19.QMessageBox::information(this, tr("Fortune Client"),
20.tr("The following error occurred: %1.")
21..arg(tcpSocket->errorString()));
22.}
23.getFortuneButton->setEnabled(true);
24.}
```

وهي تستقبل اشارات الأخطاء.
وتحدث هذه الأخطاء اثناء اتصالك بالخادم مثل اذا ماكان المنفذ او العنوان الذي وضعته غير صحيح او حتى بعد الإتصال مثل انقطاع الإتصال من الخادم او او او وهذه الأخطاء جميعها مجملة في **Qt assistant** من قائمة الادوات المجملة في **Qt** وهي موجودة في ثابت رقمي واحد وهو **socketError**.
في هذه الدالة قمنا بإستقبال هذه الأخطاء لنحدد نوعها وترسل رسالة بعد ذلك لتحدد نوع الخطأ عبر إستخدام الكلمة المفتاحية **switch** ومن ثم كان من الضروري إزالة القفل عن زر طلب نص جديد وذلك لسبب عدم وجود أي مهمة يقوم بها يستدعي اقفال الزر .

```
void Client::enableGetFortuneButton()
```

```
 {
     getFortuneButton->setEnabled(!hostLineEdit-
>text().isEmpty()
                                  && !portLineEdit-
>text().isEmpty());
 }
```

قمنا بإنشاء هذه الدالة لتلتقط اشارات الكتابة على مرب العنوان او مربع المنفذ فإذا كان أحدهما فارغا قمنا بجعل زر طلب النص في حالة مقفلة .

في هذا البرنامج أستخدمنا بروتوكول tcp في جلب البيانات ولكن ماذا لو أستخدمنا بروتوكول udp في التعامل معهاكيف سنستخدمها وبماذا تختلف عن tcp في استخدامها ؟
كما ذكرنا سابقا ان QabstractSocket يرث منه كلا من QtcpSocket و QudpSocket .
الصف QtcpSocket لا يحتوي على دوال بينما ان الآخر يحتوي على دوال تتعامل مع البيانات خاصة به ومستقلة فهذا ما سوف يجعل عملنا معها مختلف قليلا عن عملنا مع tcp او بإمكانك القول ان الإختلاف سوف يكون في المصطلحات فقط .
ولكن ما موضوع السرعة والجودة في أحد الأمثلة من كتاب Foundations of Qt Development قاموا بصناعة خادم ل tcp ثم قامو بعمل برنامج عميل بإستخدام بروتوكول tcp وخادم وعميل آخرين بإستخدام بروتوكول udp.
وعند إلقاء نظرة الى الصور ستجد أن الصورة التي تم جلبها بإستخدام بروتوكول tcp انقى من نظيرتها في udp القي نظرة على الصورتين .

هذه الصورة قد تم جلبها بإستخدام بروتوكول tcp

هذه الصورة قد تم جلبها بإستخدام بروتوكول udp

التعامل مع بروتوكول udp (بروتوكول وحدة بيانات المستخدم)

سوف يكون الإختلاف في التعامل عن tcp ان udp ترسل وتستقبل عن طريق مايسمى بوحدة البيانات datagram وهي بيانات مغلف ب udp .

أهم الدوال المستخدمة :

Bind الإتصال بالخادم .
hasPendingDatagrams ترجع true اذا ماكانت هناك بيانات متوفرة للقراءة .
readDatagram قراءة البيانات من الخادم الى متغير QbyteArray او char* وحجمها الى متغير رقمي .
writeDatagram وتكتب البيانات الى الخادم .

برنامجي الخادم والعميل بإستخدام بروتوكول udp .

كفكرة عامة عن البرنامجين فإن كلاهما مبنيين على برووكول udp سواء الخادم والعميل بإستخدام صف وحيد هو QudpSocket فالخادم يقوم بإرسال صورة والعميل يقوم بإستقبالها .

برنامج الخادم بإستخدام بروتوكول udp :

ويرسل بيانات الصور الى الخادم .

رأس الصف

```
1. class Sender : public Qobject
2. {
3. Q_OBJECT
4. public:
5. Sender();
6. private slots:
7. void broadcastLine();
8. private:
9. QUdpSocket *socket;
10.QImage *image;
11.};
```

غير دالة البناء فهناك مستقبل وحيد فقط للإشارات المؤقت .

```
1. Sender::Sender()
2. {
3. socket = new QUdpSocket( this );
4. image = new QImage( "test.png" );
5. if( image->isNull() ;
6. qFatal( "Failed to open test.png" );
7. QTimer *timer = new QTimer( this );
8. timer->setInterval( 250 );
9. timer->start();
```

```
10.connect( timer, SIGNAL(timeout()), this,
    SLOT(broadcastLine()) );

11.}
```

قمنا بتهيئة متغيرات ال socket و image ثم أختبرنا إن كان المتغير image فارغ فهذا يدل على أنه لم ينشئ اللوحة ويتم إرسال رسالة بعدم القدرة على فتح الصورة ثم قمنا بتهيئة المؤقت وهو يقوم بإرسال اشارة كل ربع ثانية تمر ثم جعلناه يبدأ وجعلنا الإشارات التي يرسلها تستقبلها الدالة broadcastLine .

```
1. void Sender::broadcastLine()

2. {

3. QByteArray buffer( 6+3*image->width(), 0 );

4. QDataStream stream( &buffer,
   QIODevice::WriteOnly );

5. stream.setVersion( QDataStream::Qt_4_0 );

6. stream << (quint16)image->width() <<
   (quint16)image->height();

7. quint16 y = qrand() % image->height();

8. stream << y;

9. for( int x=0; x<image->width(); ++x )

10.{

11.QRgb rgb = image->pixel( x, y );

12.stream << (quint8)qRed( rgb ) <<
   (quint8)qGreen( rgb ) << (quint8)qBlue( rgb );

13.}

14.socket->writeDatagram( buffer,
   QHostAddress::Broadcast,9988);

15.}
```

لو ألقيت نظرة سريعة اليه فسوف تلاحظ اننا قمنا في بادئ الأمر بالتصريح عن المتغير buffer الذي سوف يحفظ حجم أبعاد الصورة التي سوف ننقلها .
كل بكسل سوف يأخذ 3 بايت من الذاكرة فقمنا وضربناها ب 3 ثم اردنا ان نحفظ بعض المعلومات التي قد تستهلك 6 بايت من مساحة الذاكرة لتحصل في النهاية على حجم المعلومات التي سوف ترسلها ثم صرحنا عن متغير QdataStream وهو الذي بدوره يأخذ المتغير buffer لكي يحفظ كل مايرسل له

من معلومات ثم ننتقل الى السطر 5 لنضع إصدار Qt وأرسلنا لها في البداية إرتفاع وعرض الصورة و y وهو سوف يختار خط عشوائي من الصورة ليتم إرساله وبعملية التكرار يتم إرسال الألوان بصيغة rgb لكل بكسل في الخط وفي السطر الأخير وهو أهم ما في الموضوع بأسره قمنا بكتابة البيانات الى الخادم المرسل.

برنامج العميل بإستخدام بروتوكول udp :

ويقرأ البيانات التي تأتي من الخادم ليحولها بعد ذلك الى صورة .

رأس الصف

```
1. class Listener : public Qlabel
2. {
3. Q_OBJECT
4. public:
5. Listener( QWidget *parent=0 );
6. private slots;
7. void dataPending();
8. private:
9. QUdpSocket *socket;
10.QImage *image;
11.};
```

غير دالة البناء فهناك مستقبل إشارات وحيد مرتبط بإشارات توفر بيانات جديدة للقراءة
ومتغيرات من ظمنها QudpSocket ضع عينيك عليه .

جسم الصف

```
1. Listener::Listener( QWidget *parent ) :
   QLabel( parent )
2. {
3. setText( "Waiting for data." );
```

```
4. image = 0;
5. socket = new QUdpSocket( this );
6. socket->bind(9988 );
7. connect( socket, SIGNAL(readyRead()), this,
   SLOT(dataPending()) );
8. }
```

3-5 بما ان الصف يرث من الصف Qlabel في بادئ الأمر قمنا بوضع النص على ال label الى حين توفر بيانات الصورة لنضعها , وتم تهيئة المؤشر image بالقيمة 0 وهيأنا بعدها المتغير socket وطلبنا ربطه بالخادم بواسطة الدالة bind في حال عدم إضافة لها متغير من النوع QHostAddress فإنها تقبل أي عنوان ولكن ليس بوسعك ترك المنفذ فارغ وهو الذي عرفناه بالقيمة 9988 ثم قمنا بعملية اتصال بين إشارة توفر بيانات جديدة للقراءة ومستقبل الإشارة الذي صنعناه لقراءة البيانات .

```
1. void Listener::dataPending()
2. {
3. while( socket->hasPendingDatagrams() )
4. {
5. QByteArray buffer( socket->pendingDatagramSize(),
   0 );
6. socket->readDatagram( buffer.data(), buffer.size() );
7. QDataStream stream( buffer );
8. stream.setVersion( QDataStream::Qt_4_0 );
9. quint16 width, height, y;
10.stream >> width >> height >> y;
11.if( !image )
12.image = new QImage( width, height,
   QImage::Format_RGB32 );
13.else if( image->width() != width || image->height() !
   = height )
14.{
```

```
15.delete image;
16.image = new QImage( width, height,
   QImage::Format_RGB32 );
17.}
18.for( int x=0; x<width; ++x )
19.{
20.quint8 red, green, blue;
21.stream >> red >> green >> blue;
22.image->setPixel( x, y, qRgb( red, green, blue ) );
23.}
24.}
25.setText( "" );
26.setPixmap( QPixmap::fromImage( *image ) );
27.resize( image->size() );
28.}
```

10-3في بادئ الأمر جعلنا الأمر يحدث بتكرار في حال توفر بيانات للقراءة الخادم يقوم بإرسال البيانات على شكل سطر عشوائي تلو سطر عشوائي آخر ثم قمنا بصنع متغير QbyteArray ليتلقى البيانات ثم طلبنا بقراءت البيانات من الخادم بواسطة إستخدام readDatagram ليتم حفظ البيانات في المتغير الأول وبعدها صرحنا عن متغير من QdataStream ليقرأ هو الآخر من المتغير الأول الذي أحتفض بالبيانات buffer ثم وضعنا اصدار Qt فأنشأنا متغيرات رقمية ثلاثة لنزودها بالمعلومات من خلال قراءتنا ل ل buffer كما في السطر 10 .
11-28 ثم وضعنا شرط اذا كان المتغير mage وهو الذي يقوم بترجمة بيانات الصورة ليشكل بعد ذلك شكلها اذا كان فارغ فإنه يكمل الشرط عند الإكمال سيطلب بفتح لوحة رسم image جديدة ويعطيها الأبعاد التي قرأناها وإلا ان كانت image غير فارغة فإنه يتأكد بأن هل الأبعاد متوافقة مع الأبعاد التي قرأناها ان كانت الإجابة لا فإنه يقوم بحذفها وينشئ لوحة رسم ذات أبعاد جديدة مساوية للتي قرأناها ثم بدأ بعد ذلك في جلب البيانات الى الصورة.
من المعلوم مسبقا ان الخادم سوف يرسل بيانات متتالية لأسطر يختارها عشوائيا فإننا في هذه الحالة سوف نقوم بقراءة سطر سطر السطر الأول سوف نقوم بقراءته بكسل بكسل ثم ننتقل السطر العشوائي الثاني ونقوم بقراءته أيضا

بكسل بكسل وهكذا وهذا ماسوف تلاحظه في عملية التكرار for وفي النهاية بعد بناء الصورة قمنا بوضعها على النافذة الرئيسية .
ملاحظة : يمكنك قراءة وكتابة الصور مباشرة عن طريق QimageReader و QimageWriter ولكن هدفنا من هذا المثال كان ان البيانات قد تفشل في العبور للعميل وقد تنجح بعكس tcp الى هنا ينتهي برنامجين الخادم والعميل بإستخدام بروتوكول udp.

إضافة :

كما لاحظت من المثال السابق ووجدت أن هناك استقلال لكل من الخادم والعميل فليس من الظروري ان يكون برنامج الخادم مشتغلا حتى يعمل برنامج العميل فبإمكانك إرسال البيانات ومن ثم إقفال برنامج الخادم وستجد أن العميل يستقبلها بالرغم من ذلك .
بالرغم من البساطة والضعف في إلتقاط العميل للعناصر المرسلة الا ان الخادم من ممكن ان يساعد في هذه المهمة في حالة أي بيانات تم الفشل في ارسالها يمكن العميل ان يطلب منه ان يرسلها مرة أخرى حتى تتأكد من أنه قد أستلمها .
متا تحتاج الى tcp او إلى udp :
Tcp : سوف يكون مفيدا لك في إنشاء جلسات عمل او نقاشات بين مجموعة من الأجهزة تضمن أن معلومات سوف تصل بأمان للمتصلين فقط كاملة بدون نقص.
Udp: اذا كانت لديك حمولة كبيرة من البيانات فإن أحد وضائفه هي تغليفها بغلاف udp لتصبح مايسمى بوحدة البيانات ليتم إرسالها بعد ذلك بسرعة تفوق سرعة tcp ولكنك في النهاية لا تدري هل قد تم إرسالها ولا يدري العميل هل استقبلها كاملة ؟؟ .

مراجع

Qt assistant
Qt example
Foundations of Qt Development
C++ GUI Programming with Qt 4
مؤلف الكتاب : مطور Qt / محمد العبدلي