True Engineering

Работа с сетью

Модуль Qt Network

- C++
  - pro-файл: QT += network
  - Requires в spec: qt5-qtnetwork
- Использование в коде
  - #include <QtNetwork>
- Классы
  - QNetworkAccessManager — менеджер сетевых запросов
  - QNetworkRequest — сетевой запрос
  - QNetworkReply — ответ на сетевой запрос

Разрешения

- Internet — передача данных и подключение к Интернету
- Email — чтение писем и их отправка

QNetworkAccessManager – менеджер запросов

- QNetworkReply* *get*(const QNetworkRequest &request)
  отправить запрос методом GET
- QNetworkReply* *post*(const QNetworkRequest &request)
  отправить запрос методом POST
- QNetworkReply* *put*(const QNetworkRequest &request, …)
  отправить данные получателю запроса
- QNetworkReply* *deleteResource*(const QNetworkRequest &request)
  отправить запрос на удаление ресурса
- void *setProxy*(const QNetworkProxy &proxy)
  задать настройки прокси
- void *setCache*(QAbstractNetworkCache *cache)
  задать способ кэширования данных

QNetworkRequest – сетевой запрос

- void *setUrl*(const QUrl &url) задает URL запроса
- void *setHeader*(KnownHeaders header, const QVariant &value) задает заголовок известного типа
- void *setRawHeader*(const QByteArray &headerName, const QByteArray &headerValue) задает заголовок произвольного типа
- void *setAttribute*(Attribute code, const QVariant &value) задает атрибут запроса
- void *setPriority*(Priority priority) задает приоритет

QNetworkReplay – ответ на запрос

- QNetworkRequest *request*() const — получить объект запроса
- QUrl *url*() const — получить URL полученных или отправленных данных
- QVariant *header*(KnownHeaders header) const — получить заголовок известного типа
- QByteArray *rawHeader*(const QByteArray &headerName) const — получить заголовок произвольного типа
- QVariant *attribute*(Attribute code) const — получить атрибут запроса
- NetworkError *error*() const — получить ошибку обработки запроса
- void QIODevice::readyRead() — новые данные доступны для чтения
- void QNetworkReply::errorOccurred(QNetworkReply::NetworkError code) — обнаружена ошибка при обработке
- void QNetworkReply::sslErrors(const QList<QSslError> &errors) — обнаружены ошибки во время настройки

Пример

```
QNetworkAccessManager *manager = new QNetworkAccessManager();
QNetworkRequest request;
request.setUrl(QUrl("https://qt.io"));
request.setRawHeader("User-Agent", "MyOwnBrowser 1.0");
QNetworkReply *reply = manager->get(request);
connect(reply, SIGNAL(readyRead()), this, SLOT(slotReadyRead()));
connect(reply, SIGNAL(error(QNetworkReply::NetworkError)), this,
            SLOT(slotError(QNetworkReply::NetworkError)));
connect(reply, SIGNAL(sslErrors(QList<QSslError>)), this,          SLOT(slotSslErrors(QList<QSslError>)));
void slotReadyRead() {
   QNetworkReply *reply = qobject_cast<QNetworkReply*>(sender());
   if(reply->error()==QNetworkReply::NoError) {
      QByteArray content=reply->readAll();
      QDebug() << content;
   }
}
```

XMLHttpRequest

- open(method, url, async, user, password) — инициализация
- send() — отправить запрос
- readyState : enumeration
  - XMLHttpRequest.UNSENT — создан, но не открыт
  - XMLHttpRequest.OPENED — открыт, но не отправлен
  - XMLHttpRequest.HEADERS_RECEIVED — получены заголовки ответа
  - XMLHttpRequest.LOADING — процесс загрузки
  - XMLHttpRequest.DONE — все данные загружены
- status : int — код статуса ответа
- responseText : string — ответ в виде текста

Пример

```
function request(url, callback) {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = (function(request) {
        return function() {
        if (request.readyState === XMLHttpRequest.DONE)
            callback(request);
        };
    })(xhr);
    xhr.open("GET", url, true);
    xhr.send();
}
request("https://www.qt.io", function (request) {
    console.log(request.responseText);
});
```

QtWebSocket

- Стандарт API для протокола связи поверх TCP
- QML
  - import QtWebSockets 1.0
  - Requires в spec: qt5-qtdeclarative-import-websockets
- C++
  - pro-файл: QT += websockets
  - Requires в spec: qt5-qtwebsockets
  - PkgConfigBR в spec: Qt5WebSockets
- Использование в коде
  - #include <QtWebSockets/QtWebSockets>

Компоненты и классы Qt WebSocket

Компонент QML	**Класс C++**	**Назначение**
WebSocket	QWebSocket	TCP-сокет, основанный на протоколе WebSocket
WebSocketServer	QWebSocketServer	Сервер, основанный на протоколе WebSocket
	QMaskGenerator	Генератор 32-битных масок
	QWebSocketCorsAuthenticator	Объект авторизации для Cross Origin Requests (CORS)

Реализация сокета

- url : url — URL сервера для подключения
- active : bool — активен ли
- status : enumeration — текущее состояние
  - WebSockets.Connecting — подключается
  - WebSockets.Open — открыт
  - WebSockets.Closing — закрывается
  - WebSockets.Closed — закрыт
  - WebSockets.Error — ошибка подключения
- sendTextMessage(message) — отправить сообщение
- textMessageReceived(message) — получено сообщение
- errorString : string — сообщение об ошибке

Пример

```
WebSocket {
    id: socket
    url: "ws://echo.websocket.org"
    active: true
    onTextMessageReceived: {
        text = message
    }
    onStatusChanged: {
        if (socket.status == WebSocket.Error) {
            console.log("Error: " + socket.errorString)
        } else if (socket.status == WebSocket.Open) {
            socket.sendTextMessage("ping")
        } else if (socket.status == WebSocket.Closed) {
            text += "\nSocket closed"
        }
    }
}
```

True Engineering

True Engineering

630128, г. Новосибирск,
ул. Кутателадзе, 4г

(383) 363-33-51, 363-33-50
info@trueengineering.ru
trueengineering.ru

Новосибирский Государственный Университет