Framework

Принципы работы приложений Qt

Автор курса

Ларионенко Руслан

Опыт работы: 6 лет.

Языки программирования: C++, Python.
Направления: графический интерфейс пользователя и средне/высокоуровневая логика.

Проекты:
Xfinity: работа с графическим интерфейсом, интеграция веб обозревателя, работа с мультимедиа фреймворком.
Tivo: создание многих частей нового интерфейса Hydra, работа над функцией пропуска рекламы AdSkip.
BMV: бизнес логика поискового движка.

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Принципы работы приложений Qt

План

1. Типы приложений Qt.
2. QObject.
3. Сигналы и слоты.
4. Многопоточность.
5. Контейнеры Qt.

Типы приложений Qt

Перед использованием некоторых возможностей Qt необходимо создать специальный объект приложения:

- QCoreApplication.
- QGuiApplication.
- QApplication.

Это не является обязательным (но очень рекомендуемым) условием. Многие возможности Qt могут быть использованы без создания объектов приложений.

http://doc.qt.io/qt-5/qcoreapplication.html

QCoreApplication

Это базовый класс для всех Qt приложений. Используется для консольных приложений. Основные функции:

- Обработка событий и сигналов.
- Управление общими библиотеками (shared library) во время выполнения программы.
- Локализация приложения.
- Доступ к аргументам командной строки.

http://doc.qt.io/qt-5/qcoreapplication.html

QGuiApplication

Этот класс унаследован от QCoreApplication. Он предоставляет дополнительный функционал для создания приложений с графическим интерфейсом (Qt Quick, OpenGL, Vulkan). Основные функции:

- Настройка палитры, шрифта и стиля приложения.
- Управление окнами.
- Обработка событий оконной системы(мышь, клавиатура и т.п.).
- Доступ к буферу обмена.
- Возможность управления сессиями.

http://doc.qt.io/qt-5/qguiapplication.html

QApplication

Данный класс наследует весь функционал QGuiApplication. Также используется для приложений с графическим интерфейсом и предоставляет функционал для работы приложений, которые используют QWidget. Основные функции:

- Установка палитры, шрифта и стиля для Qwidget.
- Возможности взаимодействия с окружением рабочего стола.
- Есть доступ к размеру рабочего стола и размеру области в которой можно размещать окна.
- Нотификация об активности приложения на панели запущенных приложений.
- И др.

http://doc.qt.io/qt-5/qapplication.html

Цикл обработки событий

Для обеспечения обработки системных событий и работы сигналов и слотов нужно запустить цикл обработки событий используя метод QApplication::exec().
Данный цикл работает по следующему алгоритму:

ЕСЛИ (очередь событий (сигналов) для обработки пуста):
- *Ожидание новых событий.*
- *Обработка событий по очереди.*
- *Повторение с начала.*

При этом вызов метода QCoreApplication::exit(int returnCode) приведет к немедленному завершению цикла обработки событий.

http://doc.qt.io/qt-5/qcoreapplication.html

QObject

Класс QObject является базовым для большинства других классов во фреймворке Qt. Данный класс является основой объектной модели Qt и предоставляет доступ к большинству возможностей Qt. Для некоторых из них используется предварительная обработка исходников с помощью MOC. Возможности QObject:

- Иерархия объектов.
- Доступ к информации о реальных типах объектов без использования RTTI.
- Свойства (property), к которым можно получить доступ по имени(в виде текстовой строки).
- Макрос Q_INVOKABLE для доступа к методам объекта по их имени.
- Настройка взаимодействия между объектами через сигналы и слоты .
- "Перемещение" объектов в другие потоки.

Особенности:

- Объекты QObject не могут быть скопированы. Это сделано по причине того, что эти объекты являются сущностями (Identity), а не значениями (Value).
- При наследовании от класса QObject рекомендуется добавлять макрос Q_OBJECT после открывающей скобки класса для применения MOC к этому классу.

http://doc.qt.io/qt-5/qobject.html

Сигналы и слоты

- Сигналы - методы без реализации, которые возвращают void, помещенные в секцию signals.
- Слоты - обычные методы с реализацией, помещенные в секцию slots. Также в качестве слотов можно использовать функторы.
- После подключения сигнала одного объекта к слоту другого объекта, каждая генерация сигнала первого объекта приведет (немедленно или при следующей итерации цикла обработки событий) к вызову метода-слота второго объекта.
- Подключение сигнала к слоту осуществляется с помощью метода QObject::connect. Можно также подключать сигналы к сигналам.
- Сигнатура слота должна полностью или частично совпадать с сигнатурой сигнала.

http://doc.qt.io/qt-5/signalsandslots.html

Сигналы и слоты. Способы подключения

- AutoConnection способ подключения определяется автоматически. Qt::DirectConnection используется если получатель находится в потоке из которого был и сигнал. Иначе Qt::QueuedConnection. Тип соединения определяется во время генерации сигнала.
- DirectConnection слот вызывается немедленно после генерации сигнала и он выполняется в потоке в котором был сгенерирован сигнал.
- QueuedConnection слот выполняется когда управление возвращается в цикл событий в потоке получателя. Слот выполняется в потоке получателя.
- BlockingQueuedConnection то же самое что и Qt::QueuedConnection но выполнение потока в котором был сгенерирован сигнал приостанавливается (блокируется) пока выполнение слота не завершится. Если отправитель и получатель сигнала находятся в одном потоке, то этот поток будет заблокирован навсегда.
- UniqueConnection препятствует многократному соединению одинаковых подключений, может быть скомбинирован с другими способами подключения при помощи оператора |.

http://doc.qt.io/qt-5/signalsandslots.html

Многопоточность

Qt предоставляет несколько подходов к созданию многопоточных приложений:

- Низкоуровневый подход на основе QThread:
  - Можно передавать объекты QObject в поток QThread. Создание QThread предоставляет параллельный цикл обработки событий.
  - Наследование от QThread позволяет запустить некоторый код в другом потоке до запуска цикла обработки событий или вообще без него.
- Использование пула потоков QThreadPool и объектов задач QRunnable:
  - Позволяет повторно использовать потоки.
  - Позволяет создавать "очереди" задач.
- Высокоуровневое API QtConcurrent с использованием фьючеров QFuture:
  - Управлением потоками полностью автоматизировано и не требуется синхронизация.
  - Можно параллельно применить некоторую функцию к каждому элементу контейнера (или диапазону между двумя итераторами) или отфильтровать контейнер по заданному условию.
  - Можно передать на выполнение функцию.

http://doc.qt.io/qt-5/threads-technologies.html

Многопоточность. Синхронизация

Для синхронизации потоков Qt предоставляет следующие классы:

- QMutex и QMutexLocker.
- QReadWriteLock, QReadLocker и QWriteLocker.
- QSemaphore.
- QWaitCondition.

В приложении Qt (в том числе в потоках QThread) можно использовать средства синхронизации из стандартной библиотеки C++ (или какой-либо другой). Главное самому не запутаться:)

http://doc.qt.io/qt-5/threads-technologies.html

Контейнеры Qt

- Qt предоставляет набор классов-контейнеров, аналогичный контейнерам из библиотеки STL.
- Основными преимуществами контейнеров Qt является:
  - Потоковая безопасность (в случае доступа на чтение) за счет Implicit Sharing.
  - Поддержка итераторов в стиле Java (hasNext, next, и т.п.).
  - Оптимизация скорости, используемой памяти и размера исполняемого файла (по заверениям разработчиков).
  - Внедрение новых функций раньше чем в STL.
- Контейнеры и итераторы Qt обладают схожими с STL интерфейсами, что позволяет, например, применять алгоритмы STL к Qt итераторам.
- В Qt приложении можно использовать как Qt так и STL контейнеры, но смешивать их не рекомендуется, так как нет дешевого способа преобразования этих контейнеров между собой.

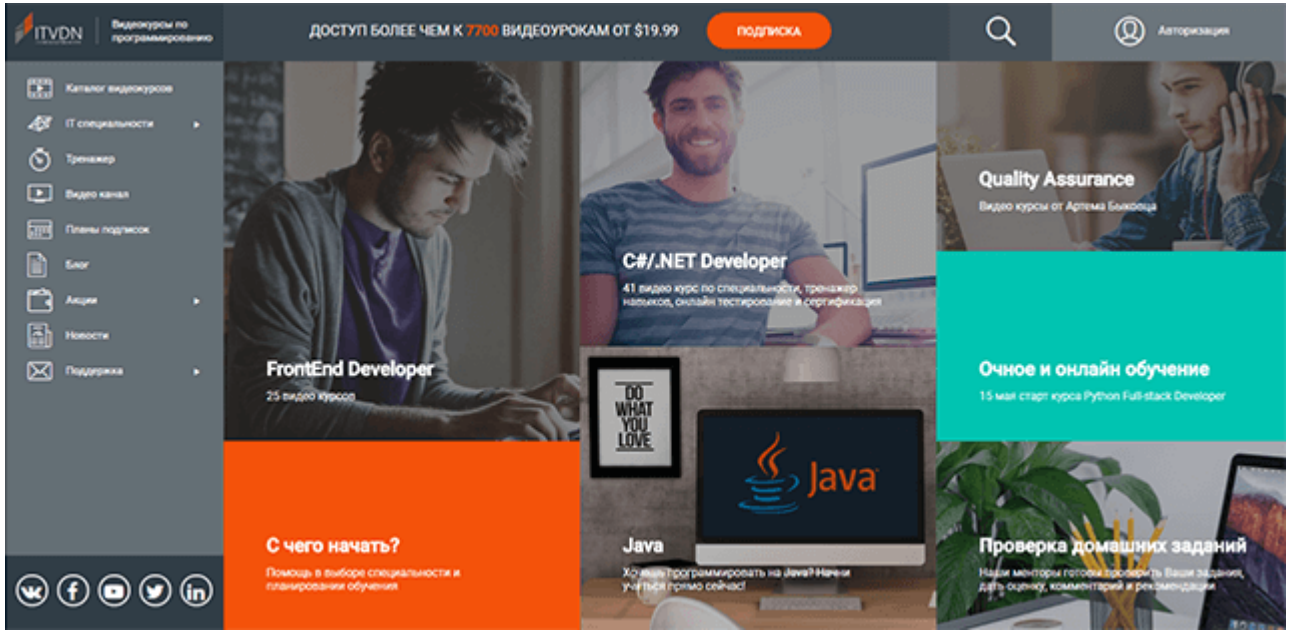
http://doc.qt.io/qt-5/containers.html

Выводы

- Qt предоставляет возможность создавать как консольные так и приложения с графическим интерфейсом.
- Система сигналов и слотов является одним из ключевых преимуществ фреймворка Qt. Данная концепция позволяет разделить систему на отдельные модули и наглядно описать взаимодействия между ними.
- Перемещение объектов в другой поток позволяет перенести некоторую часть программы в отдельный поток и безопасно с ней взаимодействовать с помочью сигналов и слотов.
- Система свойств Q_PROPERTY и методов Q_INVOKABLE позволяет использовать мета программирование (в частности рефлексию) в вашем приложении.
- Контейнеры Qt очень схожи и практически полностью совместимы с STL контейнерами.

Смотрите наши уроки в видео формате

ITVDN.com



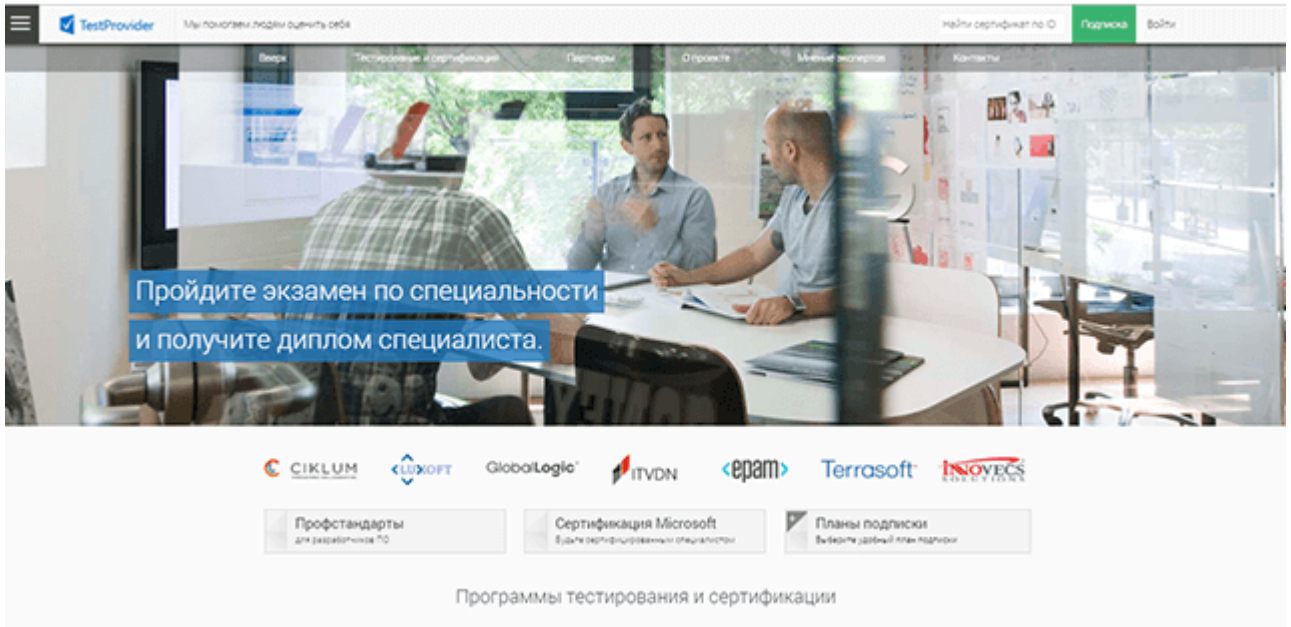
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics и другими высококвалифицированными разработчиками.

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association