АВРОРА

Модуль 1. Основы Qt Quick
Тема 1.1. Знакомство с фреймворком Qt и технологией Qt Quick
Глоссарий

Нативная разработка – это создание продукта, который пишется на оригинальных языках программирования, созданных специально для выбранной платформы. Например, родными языками для Android являются Java и Kotlin, для iOS - Swift и Objective-C. Нативное приложение будет работать только на “своей” платформе.

Кроссплатформенная разработка – это реализация приложения, которое работает на нескольких операционных системах. Это становится возможным с помощью универсального кода в кроссплатформенном фреймворке.

Фреймворк (англ. framework) - это набор библиотек, инструментов, принципов и подходов, которые предоставляют удобный функционал для разработки более сложного программного обеспечения. Он задает структуру проекта.

Qt — кроссплатформенный фреймворк, а для ОС Аврора и ряда десктопных дистрибутивов Linux на нем создаются нативные приложения.

Компиляция – это превращение программного кода в исполняемый код для процессора: на входе было то, что могли прочитать вы, а на выходе – то, что может прочитать и исполнить компьютер.

Компилятор – это программа, которая переводит текст, написанный на языке программирования, в набор машинных кодов.

Отличительная особенность – использование метаобъектного компилятора – предварительной системы обработки исходного кода. Расширение возможностей обеспечивается системой плагинов, которые возможно размещать непосредственно в панели визуального редактора.

Плагин – (от англ. plug in — подключать, plugin – подключаемый модуль). Это дополнение к базовой программе или приложению, которое расширяет её возможности и функции.

Qt работает на большом количестве разных платформ.

Библиотека разделена на ряд модулей:

- QtCore — классы ядра библиотеки, используемые другими модулями;
- Qt Quick — модуль для поддержки QML
- QtScript — классы для работы с Qt Scripts
- QtGui — компоненты графического интерфейса;
- QtNetwork — набор классов для сетевого программирования. Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами FTP и HTTP. Для работы с протоколами TCP/IP предназначены такие классы, как QTcpServer, QTcpSocket для TCP и QUdpSocket для UDP;
- QtOpenGL — набор классов для работы с OpenGL;
- QtSql — набор классов для работы с базами данных с использованием SQL. Основные классы данного модуля в версии 4.2.x: QSqlDatabase — класс для предоставления соединения с базой, для работы с

какой-нибудь конкретной базой данных требует объект, унаследованный от класса QSqlDriver — абстрактного класса, который реализуется для конкретной базы данных и может требовать для компиляции SDK базы данных. Например, для сборки драйвера под СУБД Firebird или InterBase требуются .h-файлы и библиотеки статической компоновки, входящие в комплект поставки данной СУБД;

- QtSvg — классы для отображения и работы с данными Scalable Vector Graphics (SVG);
- QtXml — модуль для работы с XML, поддерживаются модели SAX и DOM;
- QtAssistant — справочная система;
- QtTest — классы для поддержки модульного тестирования;
- Qt3Support — модуль с классами, необходимыми для совместимости с библиотекой Qt версии 3.x.x;
- QtCLucene — модуль для поддержки полнотекстового поиска, применяется в новой версии Assistant в Qt 4.4;
- QtXmlPatterns — модуль для поддержки XQuery 1.0 и Xpath 2.0;
- Phonon — модуль для поддержки воспроизведения и записи видео и аудио, как локально, так и с устройств, и по сети (Начиная с Qt 5 заменён на QtMultimedia);
- ActiveQt — модуль для работы с ActiveX и COM технологиями для Qt-разработчиков под Windows.
- QtMultimedia — модуль для поддержки воспроизведения и записи видео и аудио
- QtDeclarative — модуль, предоставляющий декларативный фреймворк для создания динамичных, настраиваемых пользовательских интерфейсов.

UI – это user interface, пользовательский интерфейс, проще говоря – оформление сайта: сочетания цветов, шрифты, иконки и кнопки.

Аббревиатура UX расшифровывается как user experience – «пользовательский опыт». Простыми словами, это то, каким образом пользователь взаимодействует с интерфейсом и насколько приложение для него понятны и удобны. В UX входит навигация по приложению, функционал меню и результат взаимодействия со страницами.

Для того, чтобы описать отличия, нужно сначала понять, что это в принципе такое. В нативных средствах IOS и Android используется **Императивный** стиль создания **UI**. То есть мы вручную создаем UI сущность, которую впоследствии мутируем различными способами. Получается мы создаём общий объект (Application), а потом поочередно меняем его поля так, чтобы он стал таким, какой нам нужен. Ещё часто говорят, что императивный способ — это о том **“как”.**

Декларативный UI Этот подход описывает **“что”** надо сделать, без ненужных деталей реализации, они все вынесены в какие-то отдельные чистые функции. Тут для того чтобы изменить UI нужно создать новый объект описания текущего состояния, просто поменять пару полей не выйдет.

QtQuick – представляет собой набор технологий для создания динамических пользовательских интерфейсов при помощи языка QML. Он разбивает пользовательский интерфейс на более мелкие элементы, которые можно объединить в компоненты. QtQuick описывает внешний вид и поведение этих элементов пользовательского интерфейса. Это описание пользовательского интерфейса можно дополнить кодом JavaScript, чтобы обеспечить не только простую, но и более сложную логику.

То есть Qt Quick представляет собой набор из нескольких технологий:

- QML - язык разметки для пользовательских интерфейсов
- JavaScript - динамический скриптовый язык
- Qt C++ - используется для внутренней реализации (back-end)

HTML, QML - это язык разметки. Он состоит из тегов, называемых типами в Qt Quick, которые заключены в фигурные скобки: *Item {}*. Он был разработан с нуля для создания пользовательских интерфейсов, ускорения и

облегчения чтения разработчиками. Пользовательский интерфейс может быть дополнительно улучшен с помощью кода JavaScript. Qt Quick легко расширяется с помощью вашей собственной встроенной функциональности с использованием Qt C++. Проще говоря, декларативный пользовательский интерфейс называется front-end, а собственные части называются back-end. Это позволяет вам отделить трудоемкую и встроенную работу вашего приложения от части пользовательского интерфейса.

Аврора – основа защищенной мобильной инфраструктуры крупных компаний.

ОС Аврора - это отечественное решение на ядре Linux, использующее в своем составе проекты с открытым исходным кодом. Также является POSIX-совместимой.

ОС Аврора — разработка российской компании «Открытая мобильная платформа». Работа над системой ведется с 2016 года.

Аврора Центр - инфраструктурное облачное решение для управления мобильными устройствами на корпоративном уровне.