Введение в QML. Основы тестирования. Работа с запросами из консоли.

Что такое QML

QML (Qt Meta-Object Language) - это декларативный язык разметки, разработанный компанией Qt для создания пользовательских интерфейсов.

Основной целью QML является упрощение процесса создания интерфейсов и легкая интеграция с кодом на C++.

Отличие QML от Qt

Чтобы лучше разобраться в том, что такое QML, давайте сравним этот язык с другим известным языком разметки — HTML.

QML и HTML представляют собой два разных языка разметки, но они имеют некоторые общие концепции. Вот как можно объяснить разницу между QML и HTML:

Контекст использования

QML применяется в основном для создания настольных и встроенных приложений, использующих фреймворк Qt.

HTML широко используется для создания веб-страниц, доступных через веб-браузеры на различных устройствах.

Поддержка функциональности

QML позволяет создавать богатые графические пользовательские интерфейсы с анимациями, эффектами и даже векторной графикой.

HTML позволяет создавать структурированный текст, изображения и интерактивные элементы, но для более сложных графических эффектов часто используются CSS и JavaScript.

Отличия QML от Qt Widgets

Qt Widgets и QML - это два различных подхода к созданию пользовательского интерфейса в приложениях, разрабатываемых с использованием фреймворка Qt. Вот основные отличия между ними:

Язык и синтаксис:

- Qt Widgets: Интерфейс создается **с использованием классов C++**, и пользовательский интерфейс определяется программно. Верстка и логика обычно разделены.
- QML: Интерфейс создается **с использованием языка разметки** QML (Qt Meta-Object Language), который представляет собой декларативный язык с похожим на JSON синтаксисом. Верстка и логика обычно интегрированы.

Графические элементы:

- Qt Widgets: Основные элементы интерфейса (например, кнопки, текстовые поля) **представлены виджетами** (widgets), которые наследуются **от классов Qt**, таких как QWidget и QLineEdit.
- QML: Графические элементы интерфейса создаются как **компоненты QML**, такие как Rectangle, Button, и Text, и они объединены в древовидную структуру.

Логика:

- Qt Widgets: Логика приложения и обработка событий обычно реализуются **с использованием классов C++ и слотов/сигналов.**
- QML: Логика часто описывается в QML с использованием JavaScript (но C++ также поддерживается), и с**обытия обрабатываются с помощью сигналов и обработчиков.**

Отличия QML от Qt Widgets

Переносимость:

- Qt Widgets: Приложения, созданные с использованием Qt Widgets, **могут быть менее переносимыми между разными платформами** и устройствами из-за более низкоуровневого характера Qt Widgets и необходимости управления различиями в стилях и размерах виджетов на разных платформах.
- QML: Qt QML часто используется для создания интерфейсов **с более высокой степенью адаптивности** и переносимости. QML легче адаптировать **под разные разрешения экранов и устройства**.

Верстка и дизайн:

- Qt Widgets: Для дизайна пользовательского интерфейса используются **стандартные средства** разработки, такие как Qt Designer.
- QML: QML обеспечивает **более гибкий и декларативный подход** к верстке интерфейса, что может облегчить создание более современных и анимированных интерфейсов.

Производительность:

- Qt Widgets: В некоторых случаях, когда требуется **высокая производительность**, Qt Widgets может быть более подходящим выбором из-за близкой интеграции с C++ и низкоуровневым управлением виджетами.
- QML: Qt QML может предоставить хорошую производительность для большинства приложений, но приложения с высокими требованиями к производительности **могут потребовать оптимизации**.

Структура QML-файла

QML-файл представляет собой текстовый файл с расширением .qml. Файл состоит из элементов интерфейса, описанных согласно декларативному синтаксису QML.

Основные элементы интерфейса в QML:

Item: Основной строительный блок интерфейса, используется для группировки и структурирования элементов.

Rectangle: Элемент для создания прямоугольных областей, которые могут содержать другие элементы.

Text: Используется для отображения текстовой информации.

Image: Позволяет отображать изображения на интерфейсе.

Button: Элемент для создания интерактивных кнопок.

ListView: Используется для отображения списков элементов с возможностью прокрутки.

MouseArea: Элемент для обработки событий мыши.

Структура QML-файла

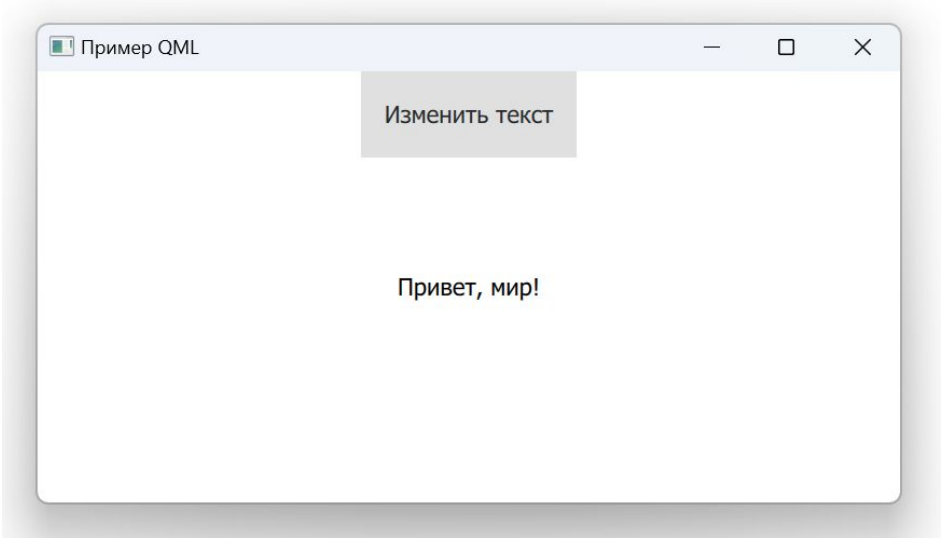
```
import QtQuick 2.12 // в ваших проектах могут быть более новые версии
import QtQuick.Controls 2.12

ApplicationWindow {
    visible: true
    width: 400
    height: 200
    title: "Пример QML"

    Rectangle {
        width: parent.width
        height: parent.height

        Text {
            id: textLabel
            text: "Привет, мир!"
            anchors.centerIn: parent
        }

        Button {
            text: "Изменить текст"
            anchors {
                horizontalCenter: parent.horizontalCenter
                top: textLabel.bottom + 10
            }
            onClicked: {
                textLabel.text = "Новый текст!"
            }
        }
    }
}
```



Основы тестирования

Для чего нужно тестирование ПО

Представьте, что мы создали новую игру или программу. Чтобы узнать, что она работает правильно и не вызывает ошибок, мы используем тестирование.

Тестирование помогает найти ошибки в программе, которые могут сделать её ненадежной или неработоспособной. Также тестирование позволяет убедиться, что программа делает всё то, что обещала делать, и не делает лишних или неправильных вещей.

Основные типы тестирования

1. Функциональное тестирование — это проверка, что программа выполняет свои задачи так, как должна. Например, кнопка "Старт" на игре действительно начинает игру.
2. Регрессионное тестирование — это когда мы проверяем, что новые изменения в программе не сломали то, что раньше уже работало. Например, после обновления игры проверяем, что старые уровни всё ещё возможно пройти без багов.
3. Пользовательское тестирование — самое массовое и естественное тестирование на обычных пользователей. Когда пользователи в отзывах говорят, что им нравится, а что нет, это помогает сделать программу удобнее.

План тестирования

План тестирования (Test Plan) — это документ, который описывает весь объем работ по тестированию и включает: описание объекта, задачи и цели тестирования, описание необходимого в процессе работы оборудования, сценарий тестирования и др.

Объект тестирования — полное описание того, что мы будем тестировать. Например, тестовыми объектами могут быть ПО для персональных компьютеров, веб-сайт или мобильные приложения.

Окружение — это конфигурация ПО, виртуального контейнера или сервера в котором планируется проводить тестирование, т.е. песочница для тестирования. Исходя из необходимого окружения, подбирается соответствующий тестовый стенд. Обычно так называют устройство (ноутбук, сервер, смартфон и планшет), на котором будут тестировать работу различных функций тестируемого ПО.

Тестовый сценарий (Test Case) — это раздел плана тестирования, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Баг-репорт

Поле	**Что содержит**
Заголовок*	Суть проблемы. Должен быть ёмким и понятным.
Описание	Обычно дублирует заголовок, поэтому не всегда обязательно.
Окружение*	Где нашли баг. Например, ОС, браузер или тестовая среда.
Шаги воспроизведения*	Описание действий, которые нужно совершить, чтобы дойти до бага.
Фактический результат*	Что видим после того, как воспроизвели баг.
Ожидаемый результат*	Что на самом деле хотели увидеть, как это должно работать по ТЗ.
Вложения	Логи, скриншоты
Дополнительная информация	Например, пояснения от тестировщика: какие способы он ещё пробовал, чтобы воспроизвести баг, и что получилось.

Работа с запросами из консоли

Попробуйте ввести команду в консоль PowerShell:

```
Invoke-WebRequest -Uri "https://drive.google.com/uc?export=download&id=1n9zg0VuhhgVLZaIeVEyZvxUdx7xSJC0J" -OutFile "вас_зарикролили.jpg"
```

Я загрузил на гугл диск картинку, таким образом её можно скачать. Подробнее о похожих запросах можете изучить самостоятельно.

Команда Invoke-WebRequest в PowerShell используется для выполнения HTTP-запросов, включая скачивание файлов с веб-серверов. Для успешного выполнения этой команды вам необходимо указать Uri (URL) ресурса, который вы хотите загрузить.