**Справочный материал**

**Содержание:**

1. **Предисловие.**
2. **Объяснение компонентов QT.**
3. **Примеры кода методов**
4. **Источники**

**Предисловие**

В этом справочном материале я постараюсь объяснить работу и структуру компонентов Qt, использующихся в этой программе. Информация взята из официальной документации Qt и переведена мной. Также я хочу внести некоторую ясность о методах и структурах, которые я использую в Creator, но не могу описать с помощью комментариев внутри IDE Qt.

**Объяснение компонентов QT**

***Q\_OBJECT***

Макрос Q\_OBJECT— сердце объектной модели QT. Главная особенность этой модели — это очень мощный механизм для беспрепятственной коммуникации между объектами, называемая сингалы и слоты (signals and slots).

***SIGNALS and SLOTS***Сигналы и слоты используются для коммуникации между объектами. Сигналы и слоты главная особенность Qt и возможно то, что больше всего отличает Qt от других сред разработки. Сигналы и слоты были созданы благодаря мета-объектной системе Qt.

В программировании графического интерфейса пользователя (GUI), когда мы меняем один виджет, мы хотим, чтобы другие виджеты были уведомлены об этом. Ещё больше мы хотим того, чтобы объекты любого вида могли коммуницировать друг с другом. Например, если пользователь кликнет на кнопку ЗАКРЫТЬ, мы вероятно хотим, что функция close() была вызвана.

Другие инструментарии достигают коммуникацию подобного рода с помощью обратного вызова (callbacks). Callback это указатель на функции, так что если вы хотите, чтобы функция обработки оповестила вас о действиях, то вы должны передать указатель на другую функцию. Затем обрабатывающая функция вызывает callback, когда это необходимо. Хотя среды разработки, которые успешно используют этот метод, существуют, callback`и могут быть не интуитивными и могут доставлять много проблем с обеспечение правильности написания аргументов обратного вызова.

В Qt мы имеем альтернативу технике callback: мы используем сигналы и слоты. Сигнал испускается, когда происходит определённое событие. Виджеты Qt имеют много встроенных (заранее определённых) сигналов, но мы всегда можем переопределить процедуру окна, чтобы добавить наши сигналы к ним. Слот — это функция, которая вызывается как ответ определённому сигналу. Виджеты Qt имеют и множество заранее определённых слотов, но привычное дело переопределить процедуру окна и добавить свои собственные слоты, чтобы вы могли их совмещать с сигналами, которые вам нужны.



Механизм сигналов и слотов безопасен: сигнатура (это сигнал, передаваемый от сигнала к слоту) сигнала должна совпадать с сигнатурой слота-получателя. (В действительности слот может получить сигнатуру короче, потому что он может игнорировать дополнительные аргументы. Поскольку сигнатуры совместимы, компилятор поможет нам заметит несостыковки, когда мы используем функцию с синтаксисом, ориентированную на указатели. Основанный на строках СИГНАЛЫ и СЛОТЫ синтаксис заметит несостыковки во время исполнения.

***СВЯЗЬ .UI (формат XML) и остального проекта***

Использование UI-файла в вашем приложении

UI-файл Qt Designer представляет собой дерево виджетов формы в формате XML. Формы могут быть обработаны:

* Во время компиляции, что означает, что формы будут преобразованы в код C++, который может быть скомпилирован.
* Во время выполнения, что означает, что формы обрабатываются классом QUiLoader, который динамически создаёт дерево виджетов при анализе XML-файла.

В НАШЕМ ПРОЕКТЕ МЫ ПОЛЬЗОВАЛИСЬ ВАРИАНТОМ 1.

## Обработка формы во время компиляции

Вы создаёте компоненты интерфейса при помощи *Qt Designer* и используете встроенные инструменты сборки **Qt**, **qmake** и **uic**, чтобы сгенерировать для них код при сборке приложения. Сгенерированный код содержит объект формы пользовательского интерфейса. Это структура C++, содержащая:

* Указатели на виджеты, компоновщики, элементы компоновщиков, группы кнопок и действия формы.
* Функцию-член setupUi() для построения дерева виджетов на родительском виджете.
* Функцию-член retranslateUi(), которая обрабатывает перевод строковых свойств формы.

Сгенерированный код может включаться в ваше приложение и использоваться непосредственно из него. Кроме того, вы можете использовать его для расширения подклассов стандартных виджетов.

Обработка формы во время компиляции может быть использована в вашем приложении одним из следующих способов:

* Прямой подход: вы создаёте виджет для использования в качестве наполнителя для компонентов и настраиваете пользовательский интерфейс внутри него.
* Подход одиночного наследования: вы создаёте подкласс базового класса формы (QWidget или QDialog, например) и подключаете закрытый экземпляр объекта пользовательского интерфейса формы.
* Подход множественного наследования: вы наследуетесь как от базового класса формы, так и от объекта пользовательского интерфейса формы. Это позволяет виджетам, определённым в форме, быть использованными напрямую в рамках подкласса.

**Примеры кода**