Actions (черновик)

Определения:

1) Действие – некоторая функциональность, выполняемая как единое целое, в рамках одного контейнера (объекта действия). Все действия являются плагинами и загружаются динамически во время запуска главного приложения.

2) Цепочка действий (или, более корректно, набор действий) – множество действий, выполняемых как одно целое в рамках одного сеанса выполнения.

3) Контекст – хранилище данных (элементов данных), доступ к которым осуществляется по строковому ключу. Помимо хранения данных, служит для взаимодействия между действиями.

4) Сеанс выполнения – временной интервал, состоящий из выбора цепочки (набора) действий, запуска ее на выполнение, ожидание окончания (успешного или нет) ее выполнения (либо ручная отмена ее выполнения), просмотра результатов работы (в случае успешного выполнения).

События действий:

1) Событие о возникновении ошибки (с сигнатурой void ErrorOccured(ExceptionData exception)) генерируется действием при возникновении ошибки и/или исключительной ситуации. При этом исключение должно быть отловлено внутри самого действия (внутри одного из обработчиков событий, принадлежащих этому действию).

2) Событие о завершении работы действия (с сигнатурой void ActionFinished()) генерируется действием при завершении его работы.

События контекста:

1) Событие об изменении данных (с сигнатурой void DataChanged(QString key)) генерируется, когда данные, расположенные в некотором элементе по ключу key были изменены. О том, что данные изменены и необходимо сгенерировать данное событие решает тот, кто изменят данные в элементе по ключу key (т.е. некоторое действие).

2) Событие об окончании изменения данных (с сигнатурой void DataCompleted(QString key)) генерируется, когда изменение данных, расположенных в некотором элементе по ключу key, закончено. О том, что изменение данных закончено и необходимо сгенерировать данное событие решает тот, кто изменят данные в элементе по ключу key (т.е. некоторое действие).

Выполнение цепочки (набора) действий. Описание:

1) Все действия цепочки (набора) действий запускаются в один момент времени.

2) Действия по поведению можно разбить на 3 группы. Действия, которые после запуска сразу же начинают что-то делать. Действия, ожидающие появления (изменения) некоторых данных в некотором элементе контекста. Действия, ожидающие окончания появления (изменения) всех данных в некотором элементе контекста.

3) Действия, которые после запуска сразу же начинают что-то делать, обычно генерируют данные для других действий (изменяя набор данных в некотором элементе контекста). При этом такие действия вполне могут находиться в ожидании других событий, отличных от событий контекста (см. выше). Так, например, к таким действиям будут относиться действия, обменивающиеся информацией через транспорт с внешним сервером (с оборудованием). Очевидно, что в любой цепочке (наборе) действий должно быть минимум одно такое действие, чтобы цепочка (набор) действий имела шанс завершить свою работу.

4) Действия, ожидающие появления (изменения) некоторых данных в некотором элементе контекста, позволяют обрабатывать данные по мере их поступления. При этом само действие, подписанное на данное событие, должно отслеживать в нужном ли элементе контекста были изменены данные (проверяя параметр key – строковый ключ элемента контекста, в котором произошло изменение данных).

5) Действия, ожидающие окончания появления (изменения) всех данных в некотором элементе контекста, позволяют обрабатывать данные только целиком (когда они полностью появятся). При этом само действие, подписанное на данное событие, должно отслеживать в нужном ли элементе контекста окончилось изменение данные (проверяя параметр key – строковый ключ элемента контекста, в котором произошло окончание изменения данных).

6) За создание элементов контекста отвечают те действия, которые изменяют данные (а точнее, добавляют новые данные) в этих элементах. Создание и получение элементов (по строковому ключу) в контексте защищено с помощью объекта класса QReadWriteLock.

7) За защиту доступа к внутреннему содержимому элементов контекста отвечают сами действия. Каждый элемент контекста содержит объекта класса QReadWriteLock, при чтении данных из элемента контекста действие всегда должно запрашивать блокировку чтения, а при изменении данных – запрашивать блокировку записи.

8) При изменении данных в некотором элементе контекста, действие должно сгенерировать несколько событий об изменении данных; после окончания изменения данных действие должно обязательно сгенерировать событие об окончании изменения данных. Действие может не генерировать события об изменении данных, а сгенерировать только событие об окончании изменения данных, если действие хочет, чтобы измененные им данные всегда обрабатывались как единое целое. В какой момент времени генерировать как события об изменении данных, так и событие об окончании изменения данных решает действие, которое эти данные изменяет.

9) Количество элементов контекста, в которых действие может изменять данные, не ограничено. Точно так же, количество элементов контекста, в которых действие ожидает изменения данных, не ограничено.

10) Когда действие заканчивает свою работу, оно должно сгенерировать событие о завершении своей работы.

11) Когда все действия, принадлежащие цепочке (набору) действий, завершают свою работу, то мы считаем, что цепочка (набор) действий завершила свою работу. После этого, можно посмотреть результат/результаты ее работы.

12) Если какое-то действие из цепочки (набора) действий сгенерировало событие о возникновении ошибки, то все действия, которые не завершили свою работу, будут остановлены. На данный момент, мы не выполняем откат результатов работы для тех действий, которые завершили свою работу.

13) Выполнение действий из цепочки (набора) действий может быть остановлено. При этом все действия, которые не завершили свою работу, будут остановлены. На данный момент, мы не выполняем откат результатов работы для тех действий, которые завершили свою работу.

14) Каждое действие содержит метод (точку входа) для очистки в случае остановки действия (например, отсылка специального сообщения через транспорт об остановке работы). Действие останавливается в том случае, когда оно не завершило свою работу и выполнение действий из цепочки (набор) действий останавливается или одно из действий сгенерировало событие о возникновении ошибки.

Примечание:

1) Среда времени выполнения (Qt) не гарантирует того, что сообщения, посланные некоторому объекту в некотором потоке в некотором порядке, будут получены этим объектом в том же порядке. Так, например, действие, изменяющее данные, генерирует сначала событие об изменении данных, а затем событие об окончании изменения данных, а действие, ожидающее изменения данных, может сначала получить событие об окончании изменения данных, а уже затем событие об изменении данных. Нужно быть готов к такому и уметь корректно обрабатывать такие ситуации.