Signals + Threads: Qt vs. Boost

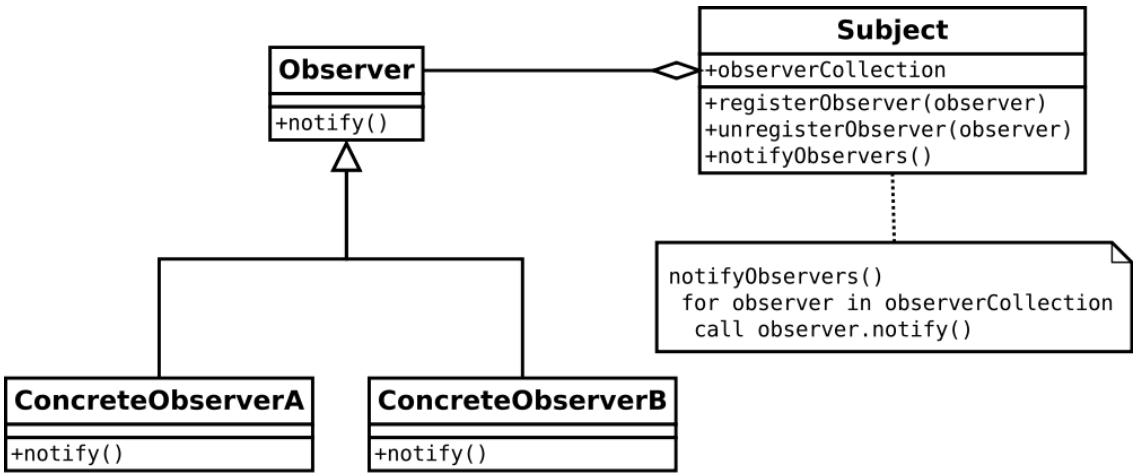
Adam Bujalski

2014-04-01

Spis Treści

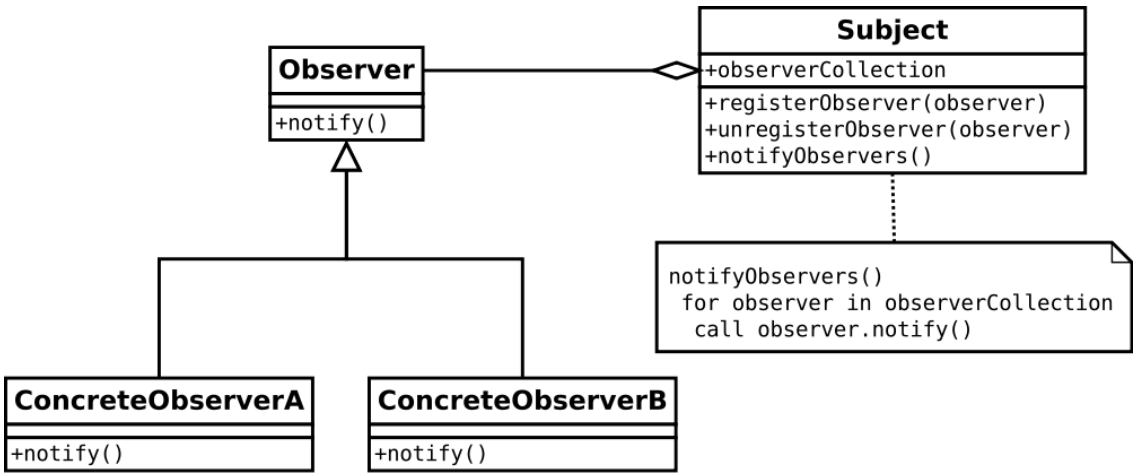
Problem w ogólnośći

- Obie biblioteki stanowią implementację wzorca obserwator



Problem w ogólnośći

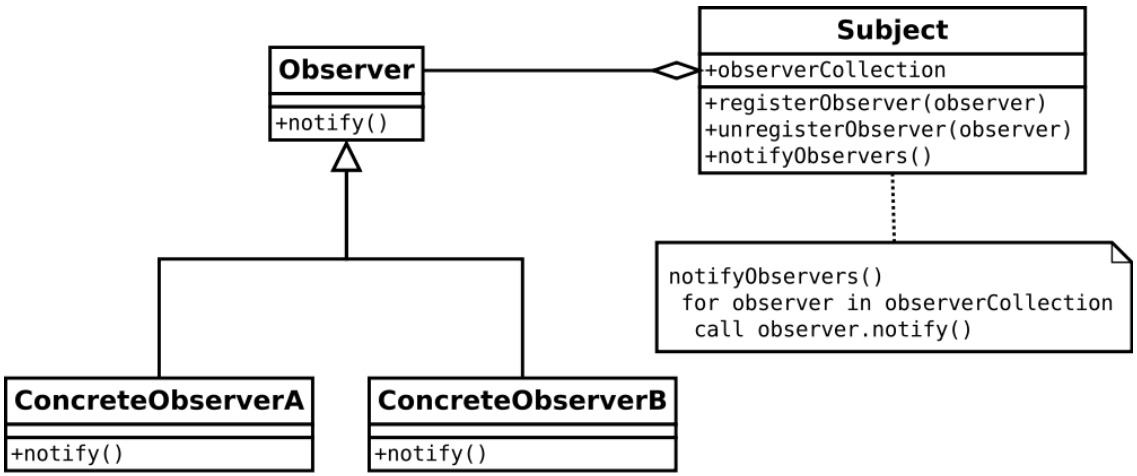
- Obie biblioteki stanowią implementację wzorca obserwator



- Obserwator $\rightarrow$ Slot
- Obserwowany $\rightarrow$ Sygnał

Problem w ogólnośći

- Obie biblioteki stanowią implementację wzorca obserwator



- Obserwator $\rightarrow$ Slot
- Obserwowany $\rightarrow$ Sygnał
- Obie biblioteki umożliwiają połączenia wiele do wiele

Łączenie sygnałów - Qt

Qt4

```
QObject::connect(&a, SIGNAL(aSignal(int)),
                 &d, SLOT(bSlot(int)));

d.connect(&a, SIGNAL(aSignal(int)), SLOT(dSlot(int)));
```

Łączenie sygnałów - Qt

Qt4

```
QObject::connect(&a, SIGNAL(aSignal(int)),
                 &d, SLOT(bSlot(int)));

d.connect(&a, SIGNAL(aSignal(int)), SLOT(dSlot(int)));
```

```
// Koniecznie w pliku naglowkowym
class A : public QObject {
signals:
    aSignal(int);    // bez ciala metody
};

class D : public QObject {
public slots:
    dSlot(int);
};
```

Łączenie sygnałów - Qt

Qt5

```
QObject::connect(&a, &A::aSignal(int),
                 &d, &D::dSlot(int));

void someFunction();
QObject::connect(button, &QPushButton::clicked,
  this, someFunction, Qt::QueuedConnection);
```

Łączenie sygnałów - Qt

Qt5

```
QObject::connect(&a, &A::aSignal(int),
                 &d, &D::dSlot(int));

void someFunction();
QObject::connect(button, &QPushButton::clicked,
  this, someFunction, Qt::QueuedConnection);
```

- Dodatkowo w Qt5 zmienił się typ zwracny przez metody `QObject::connect`

Łączenie sygnałów - Qt

Qt5

```
QObject::connect(&a, &A::aSignal(int),
                 &d, &D::dSlot(int));

void someFunction();
QObject::connect(button, &QPushButton::clicked,
  this, someFunction, Qt::QueuedConnection);
```

- Dodatkowo w Qt5 zmienił się typ zwracny przez metody `QObject::connect`
- W Qt4 był to **bool**

Łączenie sygnałów - Qt

Qt5

```
QObject::connect(&a, &A::aSignal(int),
                 &d, &D::dSlot(int));

void someFunction();
QObject::connect(button, &QPushButton::clicked,
  this, someFunction, Qt::QueuedConnection);
```

- Dodatkowo w Qt5 zmienił się typ zwracny przez metody `QObject::connect`
- W Qt4 był to **bool**
- W Qt5 jest to **QMetaObject::Connection**

Typy połączeń

- Enum `Qt::ConnectionType` określa sposób w jaki zostanie wywołany slot obsługujący dany sygnał.

Typy połączeń

- Enum `Qt::ConnectionType` określa sposób w jaki zostanie wywołany slot obsługujący dany sygnał.
- Możliwe są następujące wartości:

Stała	Wartość
`Qt::AutoConnection`	0
`Qt::DirectConnection`	1
`Qt::QueuedConnection`	2
`Qt::BlockingQueuedConnection`	3
`Qt::UniqueConnection`	0x80

Sygnały w boost

```
void slotFun(int);
struct CSlot {
    void operator() (int);
    void aSlot(int);
} aSlot;
```

Sygnały w boost

```
void slotFun(int);
struct CSlot {
    void operator() (int);
    void aSlot(int);
} aSlot;

boost::signals2::signal<void (int)> aSig;

boost::signals2::connection c = aSig.connect(slotFun);
```

Sygnały w boost

```
void slotFun(int);
struct CSlot {
    void operator() (int);
    void aSlot(int);
} aSlot;

boost::signals2::signal<void (int)> aSig;

boost::signals2::connection c = aSig.connect(slotFun);

aSig.connect(aSlot); // aSlot jest kopiowany
aSig.connect(boost::bind(&CSlot::aSlot, &aSlot, _1));
```

Emisja sygnału

Emisja sygnału

Qt

```
emit voidSignal();
emit aSignal(25);
```

Emisja sygnału

Qt

```
emit voidSignal();
emit aSignal(25);
```

boost

```
voidSignal();
aSig(25);
```

'combiner' w boost

'combiner' w boost

- W Qt wartości zwracane przez sloty wywołane podczas obsługi sygnału są ignorowane.

'combiner' w boost

- W Qt wartości zwracane przez sloty wywołane podczas obsługi sygnału są ignorowane.
- W boost można podać jako argument sygnału combiner, który umożliwia dowolne połączenie wartości zwracanych przez sloty.

'combiner' w boost

- W Qt wartości zwracane przez sloty wywołane podczas obsługi sygnału są ignorowane.
- W boost można podać jako argument sygnału combiner, który umożliwia dowolne połączenie wartości zwracanych przez sloty.
- Domyślnie **boost::optional** z wartością zwróconą przez ostatni wywoływany slot.

Przykładowy combiner

```
// aggregate_values is a combiner which places all
// the values returned from slots into a container
template<typename Container>
struct aggregate_values {
  template<typename InputIterator>
  Container operator()(InputIterator first,
        InputIterator last) const {
    Container values;
    for (; first != last; ++first)
        values.push_back(*first); // *first wywola slot
    return values;
  }
};
```

Przykładowy combiner

```
// aggregate_values is a combiner which places all
// the values returned from slots into a container
template<typename Container>
struct aggregate_values {
  template<typename InputIterator>
  Container operator()(InputIterator first,
        InputIterator last) const {
    Container values;
    for (; first != last; ++first)
        values.push_back(*first); // *first wywola slot
    return values;
  }
};

boost::signals2::signal<float (float, float),
    aggregate_values<std::vector<float> > > sig;
```

Przykładowy combiner

```
// aggregate_values is a combiner which places all
// the values returned from slots into a container
template<typename Container>
struct aggregate_values {
  template<typename InputIterator>
  Container operator()(InputIterator first,
        InputIterator last) const {
    Container values;
    for (; first != last; ++first)
        values.push_back(*first); // *first wywola slot
    return values;
  }
};

boost::signals2::signal<float (float, float),
    aggregate_values<std::vector<float> > > sig;

std::vector<float> results = sig(5, 3);
```

Spis Treści

Wątki w boost - wynik programu

```
[7f251f24e700] Hello from thread()
[7f251f250780] Hello from run_thread1()
[7f251f250780] Hello from run_thread2()
[7f251f24e700] Hello from thread()
```

Wątki w boost - program

```
#define LOG std::cout << "[" \
    << boost::this_thread::get_id() \
    << "] "

void thread()
{
    LOG << "Hello from thread()" << std::endl;
}

void run_thread1()
{
    boost::thread t(thread);
    t.join();
    LOG << "Hello from run_thread1()" << std::endl;
}
```

Wątki w boost - program c.d.

```
void run_thread2()
{
    boost::thread t(thread);
    // wg. dokumentacji jest thread_joiner
    // ale w 1.54 nie ma takiej klasy...
    boost::thread_guard<> g(t);
    LOG << "Hello from run_thread2()" << std::endl;
}

int main()
{
    run_thread1();
    run_thread2();
    return 0;
}
```

Wątki w Qt - hello_thread.h

```
#include <QThread>
#include <QDebug>
#include <iostream>

#define LOG qDebug() << "[" \
    << QThread::currentThread() << "]"

class HelloThread : public QThread {
    Q_OBJECT

protected:
    virtual void run() {
        LOG << "HelloThread";
    }
};
```

Wątki w Qt - program główny

```
#include <QCoreApplication>
#include <hello_thread.h>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    HelloThread t;
    t.start();
    LOG << "GUI/Main Thread";
    t.wait();

    return 0;
}
```

Wątki w Qt - wyjście programu

```
[ QThread(0x9fe1a0) ] GUI/Main Thread
[ HelloThread(0x7fff646c9ed0) ] HelloThread
```

Synchronizacja - prymitywy

Synchronizacja - prymitywy

- Obie biblioteki udostępniają podobny zestaw prymitywów do synchronizacji

Synchronizacja - prymitywy

- Obie biblioteki udostępniają podobny zestaw prymitywów do synchronizacji

boost	**Qt**
`mutex`	`QMutex`
`timed_mutex`	—
`recursive_mutex`	`QMutex(QMutex::Recursive)`
`recursive_timex_mutex`	—
`shared_mutex`	`QReadWriteLock`
`upgradable_mutex`	—
—	`QSemaphore`
`condition_variable[_any]`	`QWaitCondition`

Synchronizacja - prymitywy

- Obie biblioteki udostępniają podobny zestaw prymitywów do synchronizacji

boost	**Qt**
`mutex`	`QMutex`
`timed_mutex`	—
`recursive_mutex`	`QMutex(QMutex::Recursive)`
`recursive_timex_mutex`	—
`shared_mutex`	`QReadWriteLock`
`upgradable_mutex`	—
—	`QSemaphore`
`condition_variable[_any]`	`QWaitCondition`

- Dodatkowo w obu bibliotekach dostępne są mechanizmy RAII do odpowiedniego zajmownaia zasobów.

Wątki w Qt - inne opcje

Wątki w Qt - inne opcje

- W Qt można korzystać z wielowątkowości wykorzystując dodatkowe mechanizmy, które w bardziej automatyczny sposób zarządają wątkami,

Wątki w Qt - inne opcje

- W Qt można korzystać z wielowątkowości wykorzystując dodatkowe mechanizmy, które w bardziej automatyczny sposób zarządają wątkami,
- Mechanizm **QThreadPool** w połączeniu z **QRunnable**

Wątki w Qt - inne opcje

- W Qt można korzystać z wielowątkowości wykorzystując dodatkowe mechanizmy, które w bardziej automatyczny sposób zarządają wątkami,
- Mechanizm **QThreadPool** w połączeniu z **QRunnable**
- Ale ciągle wymaga on ręcznego ustalania ilości wątków do uruchomienia,

Wątki w Qt - inne opcje

- W Qt można korzystać z wielowątkowości wykorzystując dodatkowe mechanizmy, które w bardziej automatyczny sposób zarządają wątkami,
- Mechanizm **QThreadPool** w połączeniu z **QRunnable**
- Ale ciągle wymaga on ręcznego ustalania ilości wątków do uruchomienia,
- Wysokopoziomowy **QtConcurent**, który samodzielnie dostosowuje ilość wątków do ilości procesorów w systemie i udostępnia funkcyjne API MapReduce i FilterReduce.

MapReduce w Qt - map

```
typedef QMap<QString, int> WordCount;

// countWords counts the words in a single file.
// This function is called in parallel by several
// threads and must be thread safe.
WordCount countWords(const QString &file) {
  QFile f(file);
  f.open(QIODevice::ReadOnly);
  QTextStream textStream(&f);
  WordCount wordCount;

  while (textStream.atEnd() == false)
    foreach (QString word,
             textStream.readLine().split(" "))
      wordCount[word] += 1;

  return wordCount;
}
```

MapReduce w Qt - reduce

```
// reduce adds the results from map to the final result.
// This functor will only be called by one thread
// at a time.
void reduce(WordCount &result, const WordCount &w) {
    QMapIterator<QString, int> i(w);
    while (i.hasNext()) {
        i.next();
        result[i.key()] += i.value();
    }
}

QStringList files = findFiles(/* ... */);
WordCount total =
    mappedReduced(files, countWords, reduce);
```

Na koniec
Qt::QueuedConnection w boost