How to create a dictionary in Qt

This code snippet demonstrates how to create a *dictionary* of key/value pairs in Qt. The `QHash` class is a template class that provides a hash-table-based dictionary that provides very fast lookup of the value associated with a key.

`QHash` provides very similar functionality to `QMap`. Both are used to create a dictionary.

Preconditions

- Download and install the Qt SDK

Various Function of QHash class

- Returns the value associated with the key.

```
QHash<QString, int> hash;
```

- To look up a value, use operator[]() or value():

```
int num1 = hash["thirteen"];
int num2 = hash.value("thirteen");
```

- Normally, a QHash allows only one value per key. If you call insert() with a key that already exists in the QHash, the previous value is erased. For example:

```
hash.insert("plenty", 100);
hash.insert("plenty", 2000);
// hash.value("plenty") == 2000
```

Source File

```
#include <QHash>
#include <QLabel>
#include <QtGui>
#include <QApplication>
#include <QString>
#include <QHBoxLayout>
#include <QWidget>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QHash<QString, int> hash;
    QWidget win;
    QHBoxLayout *layout = new QHBoxLayout;
    hash["one"] = 1;
    hash["three"] = 3;
    hash["seven"] = 7;
    hash.insert("twelve", 12);
    QLabel *label1=new QLabel;
    int num2=2;
    int num1 = hash.value("one");
    if (hash.contains("three"))         // Checks in the hash table, Return the value if it exist
         num2 = hash.value("three");
    int num3 = hash["seven"];
    int x = hash.value("thirty", 30);    // If "thirty" is not there in a hash table than assign 30 to x.
    QString str1;
    str1.setNum(num1);
    label1->setText(str1);
    layout->addWidget(label1);
    win.setLayout(layout);
    win.show();
    return a.exec();
}
```

Similar is with QMap

QHash provides very similar functionality to QMap. The differences are:

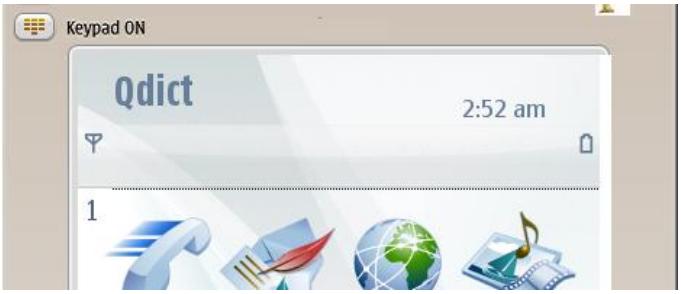
- QHash provides faster lookups than QMap. (See Algorithmic Complexity for details.)
- When iterating over a QMap, the items are always sorted by key. With QHash, the items are arbitrarily ordered.
- The key type of a QMap must provide operator<(). The key type of a QHash must provide operator==() and a global qHash(Key) function.

Source code

```
#include "Qdict.h"

#include <QtGui>
#include <QApplication>
#include <QHash>
#include<QMap>
#include <QLabel>
#include <QString>
#include <QHBoxLayout>
```


(C) Copyright Nokia 2013. All rights reserved.

```
#include <QWidget>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QMap<QString, int> map;
       QWidget win;
       QHBoxLayout *layout = new QHBoxLayout;
       map["one"] = 1;
       map["three"] = 3;
       map["seven"] = 7;
       map.insert("twelve", 12);
       QLabel *label1=new QLabel;
       int num2=2;
       int num1 = map.value("one");
       if (map.contains("three"))         // Checks in the hash table, Return the value if it exist
            num2 = map.value("three");
       int num3 = map["seven"];
       int x = map.value("thirty", 30);    // If "thirty" is not there in a hash table than assign 30 to x.
       QString str1;
       str1.setNum(num1);
       label1->setText(str1);
       layout->addWidget(label1);
       win.setLayout(layout);
       win.show();

    return a.exec();
}
```



(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.