How to convert QString to std::string?

Asked 11 years ago Active 1 year, 8 months ago Viewed 332k times

258

53

I am trying to do something like this:

```
QString string;
// do things...
std::cout << string << std::endl;
```

but the code doesn't compile. How to output the content of qstring into the console (e.g. for debugging purposes or other reasons)? How to convert `QString` to `std::string`?

c++ qt type-conversion qstring

Share Edit Follow

edited Sep 14 '15 at 3:50
Nejat
29.8k 12 96 124

asked Nov 18 '10 at 11:43
augustin
13.3k 11 58 77

10 Answers

Active Oldest Votes

309

You can use:

```
QString qs;
// do things
std::cout << qs.toStdString() << std::endl;
```

It internally uses QString::toUtf8() function to create std::string, so it's Unicode safe as well. Here's reference documentation for `QString`.

Share Edit Follow

edited Mar 7 '20 at 18:57
tunafish24
1,880 4 24 40

answered Nov 18 '10 at 11:48
Pablo Santa Cruz
167k 30 230 280

79 As of Qt 5.0, `QString::toStdString()` now uses `QString::toUtf8()` to perform the conversion, so the Unicode properties of the string will not be lost (qt-project.org/doc/qt-5.0/qtcore/qstring.html#toStdString). – Emile Cormier Apr 14 '13 at 18:13

And if you want to check the source code for `QString::toStdString` , here it is. – thuga Oct 2 '14 at 13:29

1 How can i test to see if it indeed loses the Unicode property? Will using Unicode characters(e.g. say from a language other than English) will do? – Yousuf Azad Aug 23 '16 at 3:48

▲ 242 ▼ ✓

One of the things you should remember when converting `QString` to `std::string` is the fact that `QString` is UTF-16 encoded while `std::string`... May have any encodings.

So the best would be either:

```
QString qs;

// Either this if you use UTF-8 anywhere
std::string utf8_text = qs.toUtf8().constData();

// or this if you're on Windows :-)
std::string current_locale_text = qs.toLocal8Bit().constData();
```

The suggested (accepted) method may work if you specify codec.

See: http://doc.qt.io/qt-5/qstring.html#toLatin1

Share Edit Follow

edited Mar 11 '18 at 14:57
Zhigalin - Reinstate CMs
687 1 7 32

answered Jan 10 '11 at 8:36
Artyom
29.5k 20 122 210

1 This isn't safe & is slightly slower than the proper way. You're accessing the data of a QByteArray created on the stack. The destructor for the QByteArray may be called before the constructor of the STL string. The safest way to create a helper function. `static inline std::string toUtf8(const QString& s) {    QByteArray sUtf8 = s.toUtf8();     return std::string(sUtf8.constData(), sUtf8.size()); }` – Vitali Dec 2 '11 at 2:27

18 @Vitali not correct. "The destructor for the QByteArray may be called before the constructor of the STL string" is not correct statement: Quoting the standard: 12.2.3 Temporary objects are destroyed as the last step in evaluating the full-expression (1.9) that (lexically) contains the point where they were created. And the full expression there is `std::string utf8_text = qs.toUtf8().constData();` So your statement is not correct – Artyom Dec 4 '11 at 13:55

That's true - I was thinking about const char *x = qs.ToUtf8().constData(). Still, isn't it easier to just call qs.toStdString()? – Vitali Dec 6 '11 at 14:36

6 @Vitali No. That loses non-latin1 characters. Try this: `QString s = QString::fromUtf8("árvíztűrő tükörfúrógép ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP"); std::cout << s.toStdString() << std::endl; std::cout << s.toUtf8().constData() << std::endl;` . The first is incorrect, the second is perfect. You need an utf8 terminal to test this. – Notinlist Dec 13 '11 at 22:24

3 For what it's worth, `.toStdString()` for me always results in an access violation in the pipe operator, irrespective of the `QString` 's contents (non-latin1 or not). This is on Qt 4.8.3/MSVC++ 10/Win 7. – Daniel Saner Nov 19 '12 at 12:37

36

If your ultimate aim is to get debugging messages to the console, you can use qDebug().

You can use like,

`qDebug()<<string;` which will print the contents to the *console*.

This way is better than converting it into `std::string` just for the sake of debugging messages.

Share Edit Follow

edited Mar 11 '18 at 14:57
Zhigalin - Reinstate CMs
687 1 7 32

answered Nov 18 '10 at 12:12
liaK
11.1k 10 43 70

1 qDebug() would be much better, because it supports more Qt types. – Kamil Klimek Nov 19 '10 at 13:25

25

```
QString qstr;
std::string str = qstr.toStdString();
```

However, if you're using Qt:

```
QTextStream out(stdout);
out << qstr;
```

Share Edit Follow

answered Nov 18 '10 at 11:52
chris
3,886 1 24 31

I had tried out << qstr first, before asking, but it didn't compile. It works with qstr.toStdString(), though. – augustin Nov 18 '10 at 12:38

2 I don't think so. You tried std::cout << qstr, not QTextString(stdout) << qstr; – chris Nov 18 '10 at 12:42

18

Best thing to do would be to overload operator<< yourself, so that QString can be passed as a type to any library expecting an output-able type.

```
std::ostream& operator<<(std::ostream& str, const QString& string) {
    return str << string.toStdString();
}
```

Share Edit Follow

answered Nov 18 '10 at 12:02

Puppy
140k 34 236 449

3 Why the down votes, folks? It's an overkill in my case, but who knows, it might be useful (to me or someone else). – augustin Nov 18 '10 at 12:39

I like this because Qt have a habit of changing the way their strings work - and this puts the conversion in one place. – Den-Jason Aug 28 '19 at 10:48

12

An alternative to the proposed:

```
QString qs;
std::string current_locale_text = qs.toLocal8Bit().constData();
```

could be:

```
QString qs;
std::string current_locale_text = qPrintable(qs);
```

See qPrintable documentation, a macro delivering a const char * from QtGlobal.

Share Edit Follow

edited Mar 11 '18 at 20:07

Zhigalin - Reinstate CMs
687 1 7 32

answered Sep 6 '12 at 12:09

flokk
644 8 14

2 this works even with a Qt-Build with `-no-stl` -Option set. some more info – Senči Jun 20 '13 at 13:40

8

The simplest way would be `QString::toStdString()` .

Share Edit Follow

edited May 15 '18 at 16:23
cbuchart
9,314 7 48 77

answered Aug 2 '13 at 13:21
shaveenk
1,855 4 23 36

2

You can use this;

```
QString data;
data.toStdString().c_str();
```

Share Edit Follow

answered Jan 27 '17 at 12:06
Hafsa Elif Özçiftci
173 2 12

please add some details as to what the mistake was and why your answer works
– Anantha Raju C Jan 27 '17 at 12:26

0

```
QString data;
  data.toStdString().c_str();
```

could even throw exception on VS2017 compiler in xstring

```
~basic_string() _NOEXCEPT
       {   // destroy the string
       _Tidy_deallocate();
       }
```

the right way (secure - no exception) is how is explained above from Artyom

```
QString qs;

   // Either this if you use UTF-8 anywhere
   std::string utf8_text = qs.toUtf8().constData();

   // or this if you're on Windows :-)
   std::string current_locale_text = qs.toLocal8Bit().constData();
```

Share Edit Follow

answered Aug 3 '18 at 16:54
JPM
59 5

-2

Try this:

```
#include <QDebug>
QString string;
// do things...
qDebug() << "right" << string << std::endl;
```

Share Edit Follow

edited Dec 12 '18 at 2:28
bunbun
2,546 3 30 51

answered Dec 12 '18 at 2:09
wudongliang
9

2 The question is very clear: convert QString to std::string, not to print it. – eyllanesc Dec 12 '18 at 2:58

@eyllanesc the question text says "How to output the content of qstring into the console?" , it seems OP assumes converting to std::string is the only way. It's really two questions being asked at once . – M.M Mar 9 '20 at 21:48

@M.M The question seems unclear since the question in the title says *How to convert QString to std::string?*, Maybe it's an XY problem. – eyllanesc Mar 9 '20 at 21:51