

Contents

- [1 Use Case](#)
- [2 Code](#)
- [3 Explanation](#)
- [4 Usage](#)

Use Case

We may need to filter the directory content. For example we want to list all mp3 files in a path.

QDirIterator is used to iterate through a path and filter all files with a particular suffix.

Code

The following code snippet takes a path and filters as parameters. Returns a list with all files which matched the filter

```
QStringList MyIterator::getDirContent(const QString& aPath,
                                     QStringList aFilters)
{
    // append the filtered files to this list
    QStringList list;

    // set dir iterator
    QDirIterator dirIterator(aPath,
                            aFilters,
                            QDir::AllDirs|QDir::Files|QDir::NoSymLinks,
                            QDirIterator::Subdirectories);
    while(dirIterator.hasNext())
    {
        // make sure to call next, failing todo so will result in infinite loop
        dirIterator.next();

        // found required file
        if(dirIterator.fileInfo().completeSuffix().contains(KFilterSuffix, Qt::CaseInsensi
        {
            list.append(dirIterator.fileInfo().absoluteFilePath());
        }
    }

    return list;
}
```

Explanation

```
QDirIterator dirIterator(aPath,
                        aFilters,
                        QDir::AllDirs|QDir::Files|QDir::NoSymLinks,
                        QDirIterator::Subdirectories);
```

Setup a directory iterator on the supplied path. In the second argument we set the filters (required file suffixes). Using third argument we set the Directory filters. In this case we check all directories, files and ignore symbolic links. The last argument is iterator flags which is set to iterate sub directories also.

```
dirIterator.hasNext();
```

Check if the iterator has any entries

```
// make sure to call next, failing todo so will result in infinite loop
dirIterator.next();
```

next() will increment the iterator to the next item. Make sure to call this, else the while loop becomes infinite loop.

Usage

```
QString musicPath = QDesktopServices::storageLocation(QDesktopServices::MusicLocation);
QStringList filters;
filters<<"*.mp3";
QStringList myMp3SongsList = getDirContent(musicPath, filters);
```