

```

#!/usr/bin/env python
# qlinkchecker 0.2 - a simple link checker using PyQt
# By Fredrik Fornwall <fredrikfornwall@gmail.com>

import getopt, sets, sys, time, urllib

try:
    import qt, qtnetwork, qtxml
except:
    print 'ERROR: PyQt (http://www.riverbankcomputing.co.uk/pyqt/) could not be imported'
    print '      You must have PyQt installed in order to run this program - sorry!'
    sys.exit(1)

# Safe to put it here?
doc = qtxml.QDomDocument()

class GlobalsClass:
    def __init__(self):
        self.followRedirects = False
        self.stayInDirectory = False
        self.maxRecurSIONs = 2
        self.visitExternalSites = False

        self.timeout = 0

        self.startingUrl = None          # A QUrl
        self.startingUrlString = ''      # A python string
        self.startingUrlHostString = ''  # A python string for the host
        self.visitedUrls = sets.Set()
        self.linkHandlers = []

        self.activeCount = 0

Globals = GlobalsClass()

def linkHandlerDone(linkHandler):
    "Each LinkHandler reports here when done"
    if linkHandler.done: return
    linkHandler.done = True
    Globals.activeCount -= 1
    if Globals.activeCount == 0:
        widget.startButton.setDisabled(False)
        if widget.settingsWidget.alertOnDoneCheck.isChecked():
            qt.QMessageBox.information(mainWindow, 'Link Check Done', 'All links have been checked')

class LinkHandler:
    def __init__(self, url, referer, depth, redirected = False):
        "This constructor must check that the url should indeed be visited"
        assert(url.hasHost())
        if depth > Globals.maxRecurSIONs: return

        self.urlString = url.toString()
        self.url = url

        s = str(self.urlString)
        if s in Globals.visitedUrls: return
        Globals.visitedUrls.add(s)

        Globals.activeCount += 1
        Globals.linkHandlers.append(self)

        self.depth = depth
        self.isHtml = False
        self.moreThanHeader = False
        self.http = qtnetwork.QHttp(url.host())
        self.referer = referer          # python string
        self.status = 'Waiting'
        self.parsedStatus = 'No'

```

```

self.mimeType = ''
self.done = False
self.responseTime = ''
self.redirected = redirected
self.abortTimeout = False

self.listItem = qt.QListViewItem(widget.listView, 'Waiting', self.urlString.mid(7),
self.referer.mid(7), '', '')

qt.QObject.connect(self.http, qt.SIGNAL('responseHeaderReceived(const
QHttpResponseHeader&)'), self.responseHeaderReceived)
qt.QObject.connect(self.http, qt.SIGNAL('requestFinished(int, bool)'), self.requestFinished)

self.http.get(self.url.path())

self.startTime = qt.QTime.currentTime()
if Globals.timeout > 0: qt.QTimer.singleShot(Globals.timeout, self.timeout)

def timeout(self):
if self.done or self.abortTimeout: return
self.http.abort()
self.status = 'Timeout'
self.listItem.setText(0, self.status)
linkHandlerDone(self)

def requestFinished(self, id, error):
if not self.isHtml:
self.parsedStatus = 'Not html'
self.listItem.setText(3, self.parsedStatus)

if not self.isHtml or self.depth >= Globals.maxRecursions or ((not
Globals.visitExternalSites) and str(self.url.host()) != str(Globals.startingUrl.host())):
linkHandlerDone(self)
return

if not self.url.host() == Globals.startingUrlHostString:
linkHandlerDone(self)
return

(parseOk, errorMessage, errorLine, errorColumn) = doc.setContent(self.http.readAll())

if not parseOk:
self.parsedStatus = '%s on line %s, column %s' % (errorMessage, errorLine, errorColumn)
self.listItem.setText(3, self.parsedStatus)
linkHandlerDone(self)
return

self.parsedStatus = 'Ok'
self.listItem.setText(3, self.parsedStatus)

links = doc.elementsByTagName('a')
for i in xrange(0, links.length()):
hrefAttribute = links.item(i).toElement().attributeNode('href')
if hrefAttribute.isNull():
continue
url = qt.QUrl(hrefAttribute.nodeValue())

if url.hasRef(): url.setRef(qt.QString()) # Remove trailing references (..'#link')

# QUrl uses the 'file' protocol for relative URL:s
if not url.protocol() in ['file', 'http']:
continue
url.setProtocol('http')

if not url.hasHost(): # The path is relative
url.setHost(Globals.startingUrl.host())
pathString = str(url.path())

```

```

        if pathString.find('mailto') == 0:
            continue

        if len(pathString) > 0 and pathString[0] != '/':
            pathString = str(self.url.dirPath()) + '/' + pathString
            url.setPath(pathString)

    else:
        if not Globals.visitExternalSites and str(url.host()) !=
Globals.startingUrlHostString:
            continue

        if str(url.host()) == str(Globals.startingUrl.host()) and Globals.stayInDirectory and
not url.path().startsWith(Globals.startingUrl.path()):
            continue
            LinkHandler(url, self.urlString, self.depth + 1)
            linkHandlerDone(self)

def responseHeaderReceived(self, header):
    self.abortTimeout = True
    if self.done: return
    self.responseTime = self.startTime.msecsTo(qt.QTime.currentTime())

    self.status = str(header.statusCode()) + ': ' + str(header.reasonPhrase())
    self.listItem.setText(0, self.status)

    if header.statusCode() == 200:
        if header.hasKey('Content-type'):
            self.mimeType = header.value('Content-type')
            self.isHtml = 'text/html' in self.mimeType or 'application/xhtml+xml' in
self.mimeType

            if not self.isHtml or self.depth >= Globals.maxRecurions:
                self.http.abort()
            else:
                return
        elif header.statusCode() in [301, 302]: # Moved permanently or temporarily
            if header.hasKey('Location'):
                location = str(header.value('Location'))
                self.status += ' => ' + location
                if Globals.followRedirects:
                    url = qt.QUrl(location)
                    if not url.hasHost(): url = qt.QUrl('http://' + str(self.url.host()) + location)
                    LinkHandler(url, self.urlString, self.depth, True)
            linkHandlerDone(self)

class SettingsWidget(qt.QWidget):
    def __init__(self, parent):
        qt.QWidget.__init__(self, parent)

        layout = qt.QGridLayout(self, 2, 4, 5, 5)
        layout.setAutoAdd(True)
        layout.setResizeMode(qt.QLayout.FreeResize)

        c = self.visitExternalSitesCheck = qt.QCheckBox('Visit external sites', self)
        c.setChecked(Globals.visitExternalSites)
        qt.QWhatsThis.add(c, 'Visit sites on other hosts than the host of the original site')
        qt.QObject.connect(c, qt.SIGNAL('toggled(bool)'), self.checkToggled)

        c = self.stayInDirectoryCheck = qt.QCheckBox('Stay in directory', self)
        c.setChecked(Globals.stayInDirectory)
        qt.QWhatsThis.add(c, 'Do not go below the initial directory in recursive checking')
        qt.QObject.connect(c, qt.SIGNAL('toggled(bool)'), self.checkToggled)

        qt.QLabel('Max recursions: ', self)
        self.recursiveBox = qt.QSpinBox(self)
        self.recursiveBox.setValue(Globals.maxRecurions)
        qt.QWhatsThis.add(self.recursiveBox, 'Set how many recursions. <b>Note:</b> Setting this to
a high value may lead to a very large number of sites being checked')

```

```

qt.QObject.connect(self.recursiveBox, qt.SIGNAL('valueChanged(int)'), self.checkToggled)

self.followRedirectsCheck = qt.QCheckBox('Follow redirects', self)
self.followRedirectsCheck.setChecked(Globals.followRedirects)
qt.QWhatsThis.add(self.followRedirectsCheck, 'When a linked site is reported as moved, do we
check to see that the new location is valid?')
qt.QObject.connect(self.followRedirectsCheck, qt.SIGNAL('toggled(bool)'), self.checkToggled)

self.alertOnDoneCheck = qt.QCheckBox('Alert when done', self)
self.alertOnDoneCheck.setChecked(True)
qt.QWhatsThis.add(self.alertOnDoneCheck, 'Popup a message box when all links have been
checked')

qt.QLabel('Timeout (seconds): ', self)
self.timeoutBox = qt.QSpinBox(self)
self.timeoutBox.setValue(Globals.timeout / 1000)
qt.QObject.connect(self.timeoutBox, qt.SIGNAL('valueChanged(int)'), self.checkToggled)

def checkToggled(self, on):
    Globals.followRedirects = self.followRedirectsCheck.isChecked()
    Globals.visitExternalSites = self.visitExternalSitesCheck.isChecked()
    Globals.stayInDirectory = self.stayInDirectoryCheck.isChecked()
    Globals.maxRecursions = self.recursiveBox.value() * 1000
    Globals.timeout = self.timeoutBox.value() * 1000

class Widget(qt.QWidget):
    def __init__(self, parent):
        qt.QWidget.__init__(self, parent)

        layout = qt.QGridLayout(self, 3, 1, 5, 5)
        layout.setAutoAdd(True)

        groupBox = qt.QGroupBox('Settings', self)
        groupBox.setColumns(1)
        self.settingsWidget = SettingsWidget(groupBox)

        groupBox = qt.QGroupBox('URL', self)
        groupBox.setColumns(1)
        hbox = qt.QHBox(groupBox)
        hbox.setSpacing(5)
        qt.QLabel('http://', hbox)
        self.lineEdit = qt.QLineEdit(hbox)
        self.lineEdit.setFocus()
        qt.QWhatsThis.add(self.lineEdit, 'Enter the URL of the site (eg.
www.thedomain/path/to/page.html) whose links you want to check here.')
        self.lineEdit.setText(Globals.startingUrlString[7:])
        qt.QObject.connect(self.lineEdit, qt.SIGNAL('returnPressed()'), self.start)
        self.startButton = qt.QPushButton('&Check', hbox)
        qt.QObject.connect(self.startButton, qt.SIGNAL('clicked()'), self.start)

        self.listView = qt.QListView(self)
        self.listView.setShowSortIndicator(True)
        self.listView.addColumn('Status')
        self.listView.addColumn('URL')
        self.listView.addColumn('Referer')
        self.listView.addColumn('Parse status')
        self.listView.setResizeMode(qt.QListView.AllColumns)
        self.listView.setAllColumnsShowFocus(True)
        qt.QObject.connect(self.listView, qt.SIGNAL('doubleClicked(QListViewItem*, const QPoint&,
int)'), self.listDoubleClicked)
        self.running = False

    def listDoubleClicked(self, listViewItem, point, column):
        # TODO
        pass

    def start(self):

```

```

    Globals.activeCount = 0
    for linkHandler in Globals.linkHandlers:
        del linkHandler
    Globals.startingUrl = qt.QUrl('http://' + str(self.lineEdit.text()))
    if not Globals.startingUrl.isValid() or not Globals.startingUrl.hasHost():
        qt.QMessageBox.critical(self, 'Invalid URL', 'The specied URL seems to be invalid -
correct it and try again')
        return
    Globals.visitedUrls.clear()
    self.listView.clear()
    self.startButton.setDisabled(True)
    Globals.startingUrlString = Globals.startingUrl.toString()
    Globals.startingUrlHostString = str(Globals.startingUrl.host())
    LinkHandler(Globals.startingUrl, qt.QString(), -1)

def printUsage():
    sys.stderr.write("qlinkchecker 0.2 Usage: qlinkchecker [-n] [-i] site")
    sys.exit(1)

def showAbout():
    qt.QMessageBox.information(mainWindow, 'About', 'qlinkchecker - a link checker
(http://fornwall.net/qlinkchecker/)\n\nFredrik Fornwall <fredrikfornwall@gmail.com>')

def whatsThis():
    qt.QWhatsThis.enterWhatsThisMode()

def showOptions():
    settingsWidget.show()

if __name__ == '__main__':
    (options, arguments) = getopt.getopt(sys.argv[1:], 'efr:st:')
    for key, value in options:
        if key == '-e':
            Globals.visitExternalSites = True
        elif key == '-f':
            Globals.followRedirects = True
        elif key == '-r':
            Globals.maxRecursions = int(value)
        elif key == '-s':
            Globals.stayInDirectory = True
        elif key == '-t':
            Globals.timeout = int(value) * 1000

    if len(arguments) > 1:
        print 'Too many arguments - only supply one URL that you want to check!'
        sys.exit(1)

    if len(arguments) == 1:
        if not arguments[0][0:7] == 'http://': arguments[0] = 'http://' + arguments[0]
        Globals.startingUrl = qt.QUrl(arguments[0])
        Globals.startingUrlString = Globals.startingUrl.toString()
        Globals.startingUrlHostString = str(Globals.startingUrl.host())
    else:
        Globals.startingUrlString = qt.QString()

    app = qt.QApplication(arguments)
    mainWindow = qt.QMainWindow()
    mainWindow.setCaption('qlinkchecker 0.2')
    widget = Widget(mainWindow)
    mainWindow.setCentralWidget(widget)

    fileMenu = qt.QPopupMenu(mainWindow.menuBar())
    mainWindow.menuBar().insertItem('&File', fileMenu)
    fileMenu.insertItem('E&xit', qt.qApp.quit)

    helpMenu = qt.QPopupMenu(mainWindow.menuBar())
    mainWindow.menuBar().insertItem('&Help', helpMenu)
    helpMenu.insertItem("What's &This?", whatsThis)

```

```
helpMenu.insertSeparator()  
helpMenu.insertItem('&About', showAbout)  
  
app.setMainWidget(mainWindow)  
mainWindow.show()  
  
if len(arguments) == 1: widget.start()  
  
app.exec_loop()
```