```
# Copyright (C) 2022 The Qt Company Ltd.
# SPDX-License-Identifier: LicenseRef-Qt-Commercial OR BSD-3-Clause
from __future__ import annotations

from PySide6.QtWidgets import (
    QWidget,
    QApplication,
    QMessageBox,
    QLineEdit,
    QProgressBar,
    QPushButton,
    QHBoxLayout,
    QVBoxLayout,
    QStyle,
    QFileDialog,
)
from PySide6.QtCore import QStandardPaths, QUrl, QFile, QSaveFile, QDir, QIODevice, Slot
from PySide6.QtNetwork import QNetworkReply, QNetworkRequest, QNetworkAccessManager
import sys


class DownloaderWidget(QWidget):
    """A widget to download a http file to a destination file"""

    def __init__(self, parent=None):
        super().__init__(parent)

        self.manager = QNetworkAccessManager(self)
        self.link_box = QLineEdit()
        self.dest_box = QLineEdit()
        self.progress_bar = QProgressBar()

        self.start_button = QPushButton("Start")
        self.abort_button = QPushButton("Abort")

        self.link_box.setPlaceholderText("Download Link ...")

        self._open_folder_action = self.dest_box.addAction(
            qApp.style().standardIcon(QStyle.SP_DirOpenIcon),  # noqa: F821
            QLineEdit.TrailingPosition
        )
        self._open_folder_action.triggered.connect(self.on_open_folder)

        # Â Current QFile
        self.file = None
        # Current QNetworkReply
        self.reply = None

        # Â Default http url
        self.link_box.setText(
            "http://master.qt.io/archive/qt/6.0/6.0.1/single/qt-everywhere-src-6.0.1.zip"
        )

        # Â Default destination dir
        self.dest_box.setText(
            QDir.fromNativeSeparators(
                QStandardPaths.writableLocation(QStandardPaths.DownloadLocation)
            )
        )
```

```
    # buttons bar layout
    hlayout = QHBoxLayout()
    hlayout.addStretch()
    hlayout.addWidget(self.start_button)
    hlayout.addWidget(self.abort_button)

    # main layout
    vlayout = QVBoxLayout(self)
    vlayout.addWidget(self.link_box)
    vlayout.addWidget(self.dest_box)
    vlayout.addWidget(self.progress_bar)
    vlayout.addStretch()
    vlayout.addLayout(hlayout)

    self.resize(300, 100)

    self.start_button.clicked.connect(self.on_start)
    self.abort_button.clicked.connect(self.on_abort)

@Slot()
def on_start(self):
    """When user press start button"""

    # Â http file
    url_file = QUrl(self.link_box.text())

    # destination file
    dest_path = QDir.fromNativeSeparators(self.dest_box.text().strip())
    dest_file = QDir(dest_path).filePath(url_file.fileName())

    # Ask a question if file already exists
    if QFile.exists(dest_file):
        ret = QMessageBox.question(
            self,
            "File exists",
            "Do you want to override the file ?",
            QMessageBox.Yes | QMessageBox.No,
        )
        if ret == QMessageBox.No:
            return

        QFile.remove(dest_file)

    self.start_button.setDisabled(True)
    # Create the file in write mode to append bytes
    self.file = QSaveFile(dest_file)

    if self.file.open(QIODevice.WriteOnly):

        # Start a GET HTTP request
        self.reply = self.manager.get(QNetworkRequest(url_file))
        self.reply.downloadProgress.connect(self.on_progress)
        self.reply.finished.connect(self.on_finished)
        self.reply.readyRead.connect(self.on_ready_read)
        self.reply.errorOccurred.connect(self.on_error)
    else:
        error = self.file.errorString()
        print(f"Cannot open device: {error}")
```

```
    @Slot()
    def on_abort(self):
        """When user press abort button"""
        if self.reply:
            self.reply.abort()
            self.progress_bar.setValue(0)

        if self.file:
            self.file.cancelWriting()

        self.start_button.setDisabled(False)

    @Slot()
    def on_ready_read(self):
        """ Get available bytes and store them into the file"""
        if self.reply:
            if self.reply.error() == QNetworkReply.NoError:
                self.file.write(self.reply.readAll())

    @Slot()
    def on_finished(self):
        """ Delete reply and close the file"""
        if self.reply:
            self.reply.deleteLater()

        if self.file:
            self.file.commit()

        self.start_button.setDisabled(False)

    @Slot(int, int)
    def on_progress(self, bytesReceived: int, bytesTotal: int):
        """ Update progress bar"""
        self.progress_bar.setRange(0, bytesTotal)
        self.progress_bar.setValue(bytesReceived)

    @Slot(QNetworkReply.NetworkError)
    def on_error(self, code: QNetworkReply.NetworkError):
        """ Show a message if an error happen """
        if self.reply:
            QMessageBox.warning(self, "Error Occurred", self.reply.errorString())

    @Slot()
    def on_open_folder(self):

        dir_path = QFileDialog.getExistingDirectory(
            self, "Open Directory", QDir.homePath(), QFileDialog.ShowDirsOnly
        )

        if dir_path:
            dest_dir = QDir(dir_path)
            self.dest_box.setText(QDir.fromNativeSeparators(dest_dir.path()))


if __name__ == "__main__":

    app = QApplication(sys.argv)
```

```
w = DownloaderWidget()
w.show()
sys.exit(app.exec())
```