local DataStorage = require("datastorage")

local DocSettings = require("frontend/docsettings")

local ReadHistory = require("readhistory")

local FFIUtil = require("ffi/util")

local Ftp = require("apps/cloudstorage/ftp")

local FtpApi = require("apps/cloudstorage/ftpapi")

local InfoMessage = require("ui/widget/infomessage")

local LuaSettings = require("frontend/luasettings")

local UIManager = require("ui/uimanager")

local NetworkMgr = require("ui/network/manager")

local WidgetContainer = require("ui/widget/container/widgetcontainer")

local ftp = require("socket.ftp")

local logger = require("logger")

local util = require("util")

local \_ = require("gettext")

local T = FFIUtil.template

local Send2Ebook = WidgetContainer:new{

 name = "send2ebook",

}

local initialized = false

local wifi\_enabled\_before\_action = true

local send2ebook\_config\_file = "send2ebook\_settings.lua"

local config\_key\_custom\_dl\_dir = "custom\_dl\_dir";

local default\_download\_dir\_name = "send2ebook"

local download\_dir\_path

local send2ebook\_settings

function Send2Ebook:downloadFileAndRemove(connection\_url, remote\_path, local\_download\_path)

 local url = connection\_url .. remote\_path

 local response = ftp.get(url ..";type=i")

 if response ~= nil then

 local\_download\_path = util.fixUtf8(local\_download\_path, "\_")

 local file = io.open(local\_download\_path, "w")

 file:write(response)

 file:close()

 FtpApi:delete(url)

 return 1

 else

 logger.err("Send2Ebook: Error. Invalid connection data? ")

 return 0

 end

end

-- TODO: implement as NetworkMgr:afterWifiAction with configuration options

function Send2Ebook:afterWifiAction()

 if not wifi\_enabled\_before\_action then

 NetworkMgr:promptWifiOff()

 end

end

function Send2Ebook:init()

 self.ui.menu:registerToMainMenu(self)

end

function Send2Ebook:addToMainMenu(menu\_items)

 self:lazyInitialization()

 menu\_items.send2ebook = {

 text = \_("Send2Ebook (Receiver)"),

 sub\_item\_table = {

 {

 text = \_("Download and remove from server"),

 keep\_menu\_open = true,

 callback = function()

 if not NetworkMgr:isOnline() then

 wifi\_enabled\_before\_action = false

 NetworkMgr:beforeWifiAction(self.process)

 else

 self:process()

 end

 end,

 },

 {

 text = \_("Go to download folder"),

 callback = function()

 local FileManager = require("apps/filemanager/filemanager")

 if FileManager.instance then

 FileManager.instance:reinit(download\_dir\_path)

 else

 FileManager:showFiles(download\_dir\_path)

 end

 end,

 },

 {

 text = \_("Remove read (opened) articles"),

 keep\_menu\_open = true,

 callback = self.removeReadActicles,

 },

 {

 text = \_("Set custom download directory"),

 keep\_menu\_open = true,

 callback = self.setCustomDownloadDirectory,

 },

 {

 text = \_("Settings"),

 keep\_menu\_open = true,

 callback = self.editFtpConnection,

 },

 {

 text = \_("Help"),

 keep\_menu\_open = true,

 callback = function()

 UIManager:show(InfoMessage:new{

 text = T(\_('Send2Ebook lets you send articles found on PC/Android phone to your Ebook reader (using ftp server).\n\nMore details: https://github.com/mwoz123/send2ebook\n\nDownloads to local folder: %1'), download\_dir\_path)

 })

 end,

 },

 },

 }

end

function Send2Ebook:lazyInitialization()

 if not initialized then

 logger.dbg("Send2Ebook: obtaining download folder")

 send2ebook\_settings = LuaSettings:open(("%s/%s"):format(DataStorage:getSettingsDir(), send2ebook\_config\_file))

 if send2ebook\_settings:has(config\_key\_custom\_dl\_dir) then

 download\_dir\_path = send2ebook\_settings:readSetting(config\_key\_custom\_dl\_dir)

 else

 download\_dir\_path = ("%s/%s/"):format(DataStorage:getFullDataDir(), default\_download\_dir\_name)

 end

 if not lfs.attributes(download\_dir\_path, "mode") then

 logger.dbg("Send2Ebook: Creating initial directory")

 lfs.mkdir(download\_dir\_path)

 end

 end

end

function Send2Ebook:process()

 local info = InfoMessage:new{ text = \_("Connecting…") }

 UIManager:show(info)

 logger.dbg("Send2Ebook: force repaint due to upcoming blocking calls")

 UIManager:forceRePaint()

 UIManager:close(info)

 local count = 1

 local ftp\_config = send2ebook\_settings:readSetting("ftp\_config") or {address="Please setup ftp in settings", username="", password="", folder=""}

 local connection\_url = FtpApi:generateUrl(ftp\_config.address, util.urlEncode(ftp\_config.username), util.urlEncode(ftp\_config.password))

 local ftp\_files\_table = FtpApi:listFolder(connection\_url .. ftp\_config.folder, ftp\_config.folder) --args looks strange but otherwise resonse with invalid paths

 if not ftp\_files\_table then

 info = InfoMessage:new{ text = T(\_("Could not get file list for server: %1, user: %2, folder: %3"), ftp\_config.address, ftp\_config.username, ftp\_config.folder) }

 else

 local total\_entries = table.getn(ftp\_files\_table)

 logger.dbg("Send2Ebook: total\_entries ", total\_entries)

 if total\_entries > 1 then total\_entries = total\_entries -2 end --remove result "../" (upper folder) and "./" (current folder)

 for idx, ftp\_file in ipairs(ftp\_files\_table) do

 logger.dbg("Send2Ebook: processing ftp\_file:", ftp\_file)

 --TODO recursive download folders

 if ftp\_file["type"] == "file" then

 info = InfoMessage:new{ text = T(\_("Processing %1/%2"), count, total\_entries) }

 UIManager:show(info)

 UIManager:forceRePaint()

 UIManager:close(info)

 local remote\_file\_path = ftp\_file["url"]

 logger.dbg("Send2Ebook: remote\_file\_path", remote\_file\_path)

 local local\_file\_path = download\_dir\_path .. ftp\_file["text"]

 count = count + Send2Ebook:downloadFileAndRemove(connection\_url, remote\_file\_path, local\_file\_path)

 end

 info = InfoMessage:new{ text = T(\_("Processing finished. Success: %1, failed: %2"), count -1, total\_entries +1 - count) }

 end

 end

 UIManager:show(info)

 Send2Ebook:afterWifiAction()

end

function Send2Ebook:removeReadActicles()

 logger.dbg("Send2Ebook: Removing read articles from :", download\_dir\_path)

 for entry in lfs.dir(download\_dir\_path) do

 if entry ~= "." and entry ~= ".." then

 local entry\_path = download\_dir\_path .. entry

 if DocSettings:hasSidecarFile(entry\_path) then

 local entry\_mode = lfs.attributes(entry\_path, "mode")

 if entry\_mode == "file" then

 os.remove(entry\_path)

 local sdr\_dir = DocSettings:getSidecarDir(entry\_path)

 logger.dbg("Send2Ebook: sdr dir to be removed:", sdr\_dir)

 FFIUtil.purgeDir(sdr\_dir)

 end

 end

 end

 end

 UIManager:show(InfoMessage:new{

 text = \_("All read articles removed.")

 })

end

function Send2Ebook:setCustomDownloadDirectory()

 require("ui/downloadmgr"):new{

 onConfirm = function(path)

 logger.dbg("Send2Ebook: set download directory to: ", path)

 send2ebook\_settings:saveSetting(config\_key\_custom\_dl\_dir, ("%s/"):format(path))

 send2ebook\_settings:flush()

 initialized = false

 self:lazyInitialization()

 end,

 }:chooseDir()

end

function Send2Ebook:editFtpConnection()

 local item = send2ebook\_settings:readSetting("ftp\_config") or {text="ignore this field, it's not used here;) fill rest", address="ftp://", username="",password="" , folder="/"}

 local callbackEdit = function(updated\_config, fields)

 local data = {text=fields[1], address=fields[2], username=fields[3],password=fields[4] , folder=fields[5]}

 send2ebook\_settings:saveSetting("ftp\_config", data)

 send2ebook\_settings:flush()

 initialized = false

 Send2Ebook:lazyInitialization()

 end

 Ftp:config(item, callbackEdit)

end

function Send2Ebook:onCloseDocument()

 local document\_full\_path = self.ui.document.file

 if document\_full\_path and download\_dir\_path and download\_dir\_path == string.sub(document\_full\_path, 1, string.len(download\_dir\_path)) then

 logger.dbg("Send2Ebook: document\_full\_path:", document\_full\_path)

 logger.dbg("Send2Ebook: download\_dir\_path:", download\_dir\_path)

 logger.dbg("Send2Ebook: removing Send2Ebook file from history.")

 ReadHistory:removeItemByPath(document\_full\_path)

 end

end

return Send2Ebook