



ID		Creation date	May 5, 2009
Platform	S60 3rd Edition	Tested on devices	E71, XpressMusic 5800
Category	Python	Subcategory	network

Keywords (APIs, classes, methods, functions): network programming, python

Introduction

Network programming with sockets API is powerful, allowing creation of complex applications that can connect to Internet to exchange a diversity of data. Although sockets programming with PyS60 is very similar to the standard sockets programming with C/C++, the Python syntax is simple and straightforward, creating more with few lines of code. Moreover, WiFi is frequently present in Nokia smartphones, together with other communication methods like bluetooth. In fact, bluetooth could be used as well for this task, but WiFi may be more appropriated when you want to transfer large files or blocked contents, like sis files.

In this article two complete applications using PyS60 socket API will be demonstrated. The first one (RXFile) is a server dedicated to receive files over WiFi. Basically this server will run at port 54321, waiting for incoming connections with files to be stored in memory card. The other application (TXFile) is a client for sending files over WiFi to the cited server. It allows user to choose a file from file system as well, using PyS60 user interface. These applications may be used as examples for creating related programs, inspiring developers for building new applications with PyS60 sockets API.

Both programs are mono threads and works with Python series 1.4.x and 1.9.x. It is required to create an access point beforehand, since they will be used as default access point. For server, this is specially important since its clients need to know in which address (IP) it is running. For clients, it is a way to avoid further access point selection dialogs.

Video and screenshots

In the next video it is possible to see the server and client in action. It was used an ad-hoc connection called "Nokia", with fixed IPs for client (10.0.0.1) and server (10.0.0.2).

See the [Video](#).

```

RX File
Starting server.
IP = 10.0.0.2
Port = 54321
Repository = e:\file_upload

```

Opções Sair

Server source code and sis

This code is available at [this repository](#). Sis file for Python 1.4.5 can be [downloaded here](#).

```

# -*- coding: utf-8 -*-
# Marcelo Barros de Almeida
# marcelobarrosalmeida@gmail.com
# License: GPL 3

import sys
try:
    # http://discussion.forum.nokia.com/forum/showthread.php?p=575213
    # Try to import 'btsocket' as 'socket' - ignored on versions < 1.9.x
    sys.modules['socket'] = __import__('btsocket')
except ImportError:
    pass

from appuifw import *
import socket
import os
import e32
import struct

class RxFile(object):
    """ RxFile server class
    """
    def __init__(self):
        self.lock = e32.Ao_lock()
        self.dir = "e:\\rxfile"
        if not os.path.isdir(self.dir):
            os.makedirs(self.dir)
        self.apo = None
        self.port = 54321
        self.new_line = u"\u2029"
        app.title = u"RX File"
        app.screen = "normal"
        app.menu = [(u>About", self.about)]
        self.body = Text()
        app.body = self.body
        self.lock = e32.Ao_lock()

    def recv_file(self, cs, addr):

```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
""" Given a client socket (cs), receive a new file
    and save it at self.dir
"""
data = ""
name = ""
size = 0
# waiting for file name
while True:
    n = data.find("\n")
    if n >= 0:
        name = data[:n]
        data = data[n+1:]
        break
    try:
        buf = cs.recv(1024)
    except socket.error:
        cs.close()
        return
    data = data + buf

# waiting for file size (may be useful for limits checking)
while True:
    n = data.find("\n")
    if n >= 0:
        # unpack one long (L) using big endian (>) endianness
        size = struct.unpack(">L",data[:n])[0]
        data = data[n+1:]
        break
    try:
        buf = cs.recv(1024)
    except socket.error:
        cs.close()
        return
    data = data + buf

self.body.add(u"Receiving %s (%d bytes)" % (name,size) + self.new_line)
# waiting for file contents
fname = os.path.join(self.dir,name)
f = open(fname,"wb")
n = len(data)
while True:
    f.write(data)
    n += len(data)
    if n % 100 == 0: # a mark at each 100k
        self.body.add(u".")
    try:
        data = cs.recv(1024)
    except socket.error:
        cs.close()
        return
    if not data:
        break
self.body.add(self.new_line + u"Finished." + self.new_line)
cs.close()
f.close()

def server(self,ip,port):
    """ Starts a mono thread server at ip, port
    """
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

    try:
```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
s.bind((ip,port))
except socket.error, (val,msg):
    note(u"Error %d: %s" % (val,msg),"info")
    return

s.listen(1)

while True:
    (cs,addr) = s.accept()
    self.body.add(u"Connect to %s:%d" % (addr[0],addr[1]) + self.new_line)
    self.recv_file(cs,addr)

def sel_access_point(self):
    """ Select and set the default access point.
        Return the access point object if the selection was done or None if not
    """
    aps = socket.access_points()
    if not aps:
        note(u"No access points available","error")
        return None

    ap_labels = map(lambda x: x['name'], aps)
    item = popup_menu(ap_labels,u"Access points:")
    if item is None:
        return None

    apo = socket.access_point(aps[item]['iapid'])
    socket.set_default_access_point(apo)

    return apo

def about(self):
    note(u"RX File by Marcelo Barros (marcelobarrosalmeida@gmail.com)","info")

def run(self):
    self.apo = self.sel_access_point()
    if self.apo:
        self.apo.start()
        self.body.add(u"Starting server." + self.new_line)
        self.body.add(u"IP = %s" % self.apo.ip() + self.new_line)
        self.body.add(u"Port = %d" % self.port + self.new_line)
        self.body.add(u"Repository = %s" % (self.dir) + self.new_line)
        self.server(self.apo.ip(),self.port)
        self.lock.wait()
    app.set_exit()

if __name__ == "__main__":
    app = RxFile()
    app.run()
```

Client source code and sis

This code is available at [this repository](#). Sis file for Python 1.4.5 can be [downloaded here](#).

```
# -*- coding: utf-8 -*-
# Marcelo Barros de Almeida
# marcelobarrosalmeida@gmail.com
# License: GPL 3
```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
import sys
try:
    # http://discussion.forum.nokia.com/forum/showthread.php?p=575213
    # Try to import 'btsocket' as 'socket' - ignored on versions < 1.9.x
    sys.modules['socket'] = __import__('btsocket')
except ImportError:
    pass

from appuifw import *
import socket
import os
import e32
import struct
import time
import re

class FileSel(object):
    """
    Open a selection file dialog. Returns the file selected or None.
    Initial path and regular expression for filtering file list may be provided.

    Examples:
    sel = FileSel().run()
    if sel is not None:
        ...

    sel = FileSel(mask = r"(*\.jpeg|*\.jpg|*\.png|*\.gif)").run()
    if sel is not None:
        ...

    """
    def __init__(self, init_dir = "", mask = ".*"):
        self.cur_dir = unicode(init_dir)
        if not os.path.exists(self.cur_dir):
            self.cur_dir = ""
        self.mask = mask
        self.fill_items()

    def fill_items(self):
        if self.cur_dir == u"":
            self.items = [ unicode(d + "\\") for d in e32.drive_list() ]
        else:
            entries = [ e.decode('utf-8')
                        for e in os.listdir( self.cur_dir.encode('utf-8') ) ]
            d = self.cur_dir
            dirs = [ e.upper() for e in entries
                    if os.path.isdir(os.path.join(d,e).encode('utf-8')) ]

            files = [ e.lower() for e in entries
                    if os.path.isfile(os.path.join(d,e).encode('utf-8')) ]

            files = [ f for f in files
                    if re.match(self.mask,f) ]
            dirs.sort()
            files.sort()
            dirs.insert( 0, u".." )
            self.items = dirs + files

    def run(self):
        while True:
            item = selection_list(self.items, search_field=1)
            if item is None:
```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
        return None
    f = self.items[item]
    d = os.path.abspath( os.path.join(self.cur_dir,f) )
    if os.path.isdir( d.encode('utf-8') ):
        if f == u".." and len(self.cur_dir) == 3:
            self.cur_dir = u""
        else:
            self.cur_dir = d
            self.fill_items()
    elif os.path.isfile( d.encode('utf-8') ):
        return d

class TxFile(object):
    """ TxFile client class
    """
    def __init__(self):
        self.lock = e32.Ao_lock()
        self.apo = None
        self.dir = ""
        self.port = 54321
        self.ip = ""
        self.new_line = u"\u2029"
        app.title = u"TX File"
        app.screen = "normal"
        app.menu = [(u"Send file", self.send_file),
                    (u"Set AP", self.set_ap),
                    (u>About", self.about),
                    (u"Exit", self.close_app)]
        self.body = Text()
        app.body = self.body
        self.lock = e32.Ao_lock()

    def close_app(self):
        self.lock.signal()

    def set_ap(self):
        """ Try to set an access point, return True or False to indicate the success.
        If True, sets self.apo to choosen access point.
        """
        apo = self.sel_access_point()
        if apo:
            self.apo = apo
            return True
        else:
            return False

    def send_file(self):
        """ Send a file to server
        """
        # at least one access point is necessary
        if not self.apo:
            if not self.set_ap():
                return

        # use our own IP as initial guess
        self.apo.start()
        if not self.ip:
            self.ip = self.apo.ip()

        # get server address
        ip = query(u"Server addr", "text", unicode(self.ip))
        if ip is None:
```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
        return
self.ip = ip

# get filename
full_name = FileSel(init_dir=self.dir).run()
if full_name is None:
    return

# transmitt file
full_name = full_name.encode('utf-8')
self.dir = os.path.dirname(full_name)
base_name = os.path.basename(full_name)
size = os.path.getsize(full_name)

self.body.add(u"Connecting to %s:%d ..." % (self.ip,self.port) + self.new_line)
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
try:
    s.connect((self.ip,self.port))
except socket.error, (val,msg):
    self.body.add(u"Error %d: %s" % (val,msg) + self.new_line)
    return

self.body.add(u"Sending %s (%d bytes)" % (base_name,size) + self.new_line)
f = open(full_name,"rb")
header = "%s\n" % (base_name) + struct.pack(">L",size) + "\n"
s.sendall(header)
n = 0
ta = time.time()
while True:
    data = f.read(1024)
    if not data:
        break
    s.sendall(data)
    n += len(data)
    if n % 100 == 0: # a mark at each 100k
        self.body.add(u".")
s.close()
f.close()
tb = time.time()
self.body.add(self.new_line + u"Finished (%0.2f kbytes/s)." % ((n/1024.0)/(tb-ta)) + self.new_line)

def sel_access_point(self):
    """ Select and set the default access point.
        Return the access point object if the selection was done or None if not
    """
    aps = socket.access_points()
    if not aps:
        note(u"No access points available","error")
        return None

    ap_labels = map(lambda x: x['name'], aps)
    item = popup_menu(ap_labels,u"Access points:")
    if item is None:
        return None

    apo = socket.access_point(aps[item]['iapid'])
    socket.set_default_access_point(apo)

    return apo

def about(self):
```

Transferring_files_over_WiFi_with_PyS60_-_a_client/server_example

```
note(u"TX File by Marcelo Barros (marcelobarrosalmeida@gmail.com)", "info")

def run(self):
    self.lock.wait()
    app.set_exit()

if __name__ == "__main__":
    app = TxFile()
    app.run()
```

Enhancements

Some possible enhancements:

- Better error handling for all socket calls. Just critical calls were handled here.
- Better termination, shutting down server first.
- Add multi thread, allowing both server and client in the same application.

References

Please, read the post [Network programming for PyS60 \(VII\)](#) for a better understanding about the protocol used.