FK는 포링키? 테이블과 테이블을 조인하는 키

서로 다른 테이블 간에 같은 필드가 pk(PrimaryKey)가 있으면, 연결할 목적이면 FK PK는 유니크한 일반적으로 성격을 갖지만 FK성격도 가지면 유니크한 성격을 갖지는 않음

데이터베이스 할 때 계속 열어두고 써도 되고, 필요할 때 열었다가 다쓰면 닫는 방법이 있는데 장단점이 있음.

특히 네트워크 기반일 때는 연결에 문제가 생길 수 있기 때문에 필요 할 때 열고 다 쓰면 닫기가 좋음. 보안에도 좋음

날짜 정보까지 넣어도 좋음

===========네트워크===============

IPC(Inter-Process Communication)

-process간 통신

process:실행 중인 프로그램. 독립적인 실행 단위?????????

프로세스 실행되면 – 코드영역, 데이터영역,프로세스 정보 영역이 할당됨

32비트 cpu에서는 4GB안에는 user영역, kernel영역도 있음

코드,데이터,프로세스정보는 user영역. 우리가 접근 가능

kernel영역은 os가 담당함. 우리가 접근 불가

4GB는 가상 메모리. 물리 메모리 아님

메모리운영 방식을 가상 메모리 기법을 쓰는데

프로세스가 두개 생겼을 때 data공유나 code공유를 하고 싶으면

기본적으로 프로세스는 독립공간.

다른 프로세스가 접근하는걸 함부로 허용하지 않겠다.

그래서 데이터 공유를 하기 위해 쓰는게 IPC, 코드 공유를 하기위해 쓰는게 RPC

IPC, RPC에서 언어적으로 제공하지 않고 OS가 제공함

윈도우는 멀티 프로세스 상태에서 데이터 공유를 지원하지 않음

멀티 프로세스보다는 멀티 스레딩

리눅스는 데이터 공유 지원함.

어디에서??????는 RPC를 위해 Com, COrba, EJB를 지원함? component구조. framework형태로

Com,Corba,EJB를 적용하려면 framework를 이용해야함

IPC는 file 객체, pipe객체,socket(local- 같은 os 환경 내에서)

file 하나를 여러 프로세스가 같이 쓰면 중복문제가 생김

주로 IPC는 data공유

소켓은 도구

내부에서는 IPC를 활용하면 되는데 다른 PC에 있는 데이터가 필요한 경우. os도 다를 수 있고 PC환경도 다르지만 접근해서 가져올 수 있음. 이 때 사용하는게 네트워크 통신

네트워크를 이용한 IPC

다른 PC랑 통신하려면 시스템을 인식하고 process가 뭔지도 알아야함

그래서 IP address랑 Port Number가 필요.

**IP Address는 PC식별**

**Port Number는 Process 식별**

그리고 데이터를 공유할꺼니깐 데이터 정보도 있어야함

이런 세개를 묶어서 Packet

이런 패킷들이 많으니깐 Protocol이 존재

이런식으로 보내면 이런식으로 해석해라 하고 미리 정해놓은 약속

지금 현재 네트워크 통신 할 때 많이 쓰는게 TCP/IP라는 프로토콜을 많이씀

TCP/IP는 4계층씀 presentation – application – Network-????

인터넷에는 application – Transport- Internet – network access

TCP/IP구조

TCP Header-IP-Data-TCP footer????????

TCP/IP기반 프로토콜 HTTP, FTP, SMTP 이런 것들이 있는데 이런 것들은 주로 Presentation layer

telnet도 TCP/IP

Socket: 동일 PC/Network에 연결된 PC의 Process간에 IPC를 가능하도록 연결 시켜주는 목적. O/S가 제공

다른 시스템간에 IPC가능하도록 연결해주는게 Socket.

유닉스시절부터 사용하기 시작한 거의 표준, 예전 BSD(버클리에서 만든 유닉스)에서부터 소켓이라는 개념이 나옴

소켓을 연결하는게 1차적인 과정. 소켓을 이용해서 다른 PC간에 연결 구성을 하는거 소켓프로그램. 네트워크 프로그램을 하는거. 소켓을 이용한 프로그램을 하는거. 파이썬도 이런걸 지우너함. 언어와 상관 없이 OS상관 없이 거의 유사함

네트워크 프로그래밍 - socket이용

Server(Client 요청에 대한 응답)

1. socket 생성
2. socket 설정
3. client연결 요청, 수신 대기
4. client연결시 처리
5. 소켓소멸

Client(Server에 요청)

1. Socket 생성
2. Socket설정
3. Server에 연결
4. Server 연결실 처리
5. Socket 소멸

Server와 Client를 소켓으로 연결

언어에 상관없이 소켓을 이용하는 프로그래밍은 거의 유사

단 어떤 식으로 소켓을 추상화 해놨느냐만 다름

서버 프로그램을 어떤 식으로 구현 할건지

그리고 프로토콜을 어떻게 할건지. 이미 알려진걸 쓸건지, 내가 만들걸 쓸건지.

이미 알려진건 형식대로만 써야해서 내 임의대로 바꿀 여지가 별로 없음

IPC:같은 컴퓨터 내의 Process간 통신

Socket 통신(Network Program) : 다른 컴퓨터 내의 Process간 통신

결국 이 두개는 IPC(Process간 통신, Process간 Data공유)

소켓 하나는 단방향 통신. ???????????????????????????????

양방향 가능하지만 송수신데이터가 섞여버리는 문제가 발생함

서버의 기본 역할은 응답대기. 요청 대기

클라이언트의 기본 역할은 요청

포트번호는 해당 프로세스 내에 부여된 번호

서버 프로그램은 하나의 포트번호로 부여되어있어야 혼란이 없음. 유일 해야함

클라이언트의 포트는 중요하지 않음.

라우터의 역할은 해당 네트워크에 연결하게 해주는 역할?

우리가 활용가능한 포트 번호는 5000번 이후부터 65535 사이의 번호 사용

cmd에서 netstat -a

127.0.0.1은 루프백 – 자기자신(local host)

상태가 LISTENING인거는 대기 상태. 클라이언트로부터 요청이 들어오는걸 대기 상태

로컬주소에서:뒤에 붙은게 포트 번호

80은 http

python에서 포트번호 확인

>>> import socket

>>> socket.getservbyname('http','tcp') #tcp기반 http의 포트번호

80

>>> socket.getservbyname('ftp','tcp')

21

서버는 클라이언트로부터 요청이 없으면 아무것도 안하는게 일반겆ㄱ

from socket import \*

def server1():

svrsock=socket(AF\_INET,SOCK\_STREAM) #첫번째꺼는 소켓 타입 AF\_INET은 네트워크. 두번째꺼는 프로토콜 타입. SOCK\_STREAM은 TCP

svrsock.bind(('localhost',8000)) #서버 정보 설정. 소켓의 서버 IP주소랑 포트번호

print('Server Info: localhost'+str(8000))

svrsock.listen(1) #클라이언트 연결 대기 상태로 listen 클라이언트 연결될 때까지 대기

conn,addr=svrsock.accept() #클라이언트가 연결이 되면 accept호출이 됨 클라이언트 소켓 리턴해줌 ip랑 포트번호. 클라이언트 소켓이 conn. addr이 클라이언트 ip주소와 포트 번호

print('Client Info:'+addr[0]+':'+str(addr[1])) #addr[0]은 ip, addr[1]는 포트번호

conn.recv(1024) # 리시브 함수가 클라이언트가 보낸걸 수신하는? 수신 데이터 바이트 크기

conn.close() # 클라이언트 소켓 닫음

if \_\_name\_\_=='\_\_main\_\_':

server1()

from socket import \*

def client1():

clientsocket=socket(AF\_INET,SOCK\_STREAM)

print('Client :')

clientsocket.connect(('localhost',8000)) #서버에 접속하겠다 서버 ip랑 포트

clientsocket.send('Hello'.encode()) #send가 통신 할 떄 데이터를 보낼 때 encode는 binary

clientsocket.close() #클라이언트 소켓 닫음

if \_\_name\_\_=='\_\_main\_\_':

client1()

def server2():

HOST='localhost'

PORT=50007

s=socket(AF\_INET,SOCK\_STREAM)

s.bind((HOST,PORT))

s.listen(1)

conn, addr=s.accept()

print('Connected by',addr)

while True:

data= conn.recv(1024)

if not data: break

conn.sendall(data)

conn.close()

def client2():

HOST='localhost'

PORT=50007

s=socket(AF\_INET,SOCK\_STREAM)

s.connect((HOST,PORT))

s.sendall('Hello, world'.encode())

data=s.recv(1024)

s.close()

print('Received', repr(data))

이런게 에코서버 메시지를 받아서 다시 돌려주는거

|  |
| --- |
| Server(server.py) |
| Socket 생성 (server socket) |
| Socket.bind() |
| Socket.listen() 클라이언트에서 socket.connect()하면 여기로 옴 |
| Socket. accept() (client socket return) |
| Client와 통신 (client socket 사용 |
| Client와 통신연결 해제 |

|  |
| --- |
| Client(client.py) |
| socket생성 |
| Socket.connect()(server socket) |
| Server와 통신 |
| Server와 통신 연결 해제 |

TCP에서는 데이터의 끝이 어딘지 알기 어려움. packet의 경계(끝)정보 확인이 어렵다

Packet 형태 –

1. 고정크기

2. 가변크기

2-1 크기+내용

2-2 마지막에 경계값 설정

패킷 하나의 크기가 명확하지 않아서 패킷 구성을 해줘야함

한번 메시지 전송 할 때 고정해서 보내면 그 크기만큼의 데이터가 다 들어올 때까지 기다림

항상 일정크기가 돼야함. 그 크기만큼 채워져 있다는 보장이 없어서 낭비가 있을 수 있음.

가변크기에서는 낭비는 없는데 문제는 크기가 얼마인지 모름

그래서 크기 + 내용으로 보내던가 마지막에 경계 값 설정 해야 함

크기+내용은 두번 읽어야함. 크기랑 내용으로. 주로 binary

경계값 설정해주는건 주로 text.

패킷 형태 -> protocol

어떻게 넘어오는지 프로토콜에 구성되어있음

1. 단일 처리 서버 : 한번에 하나의 client와 통신
2. 다중 처리 서버: 동시에 여러 client와 통신
   1. 멀티 프로세싱
   2. 멀티 스레딩
   3. 멀티 플렉싱

========================GUI================================

Tkinter는 안이쁜데 내장되어있어서 바로 쓸 수 있음

Tcl은 tool command language. 프로그래밍 언어

Tk는 툴킷

xwindow

Qt나 Tk로많이 함???????????

tk객체 생성하면 메인 윈도우 생성됨

그 위에 위젯들 생성해서 배치

그리고 메인 루프 실행

GUI(Window) Program 방식-> Event 발생시 처리 화면상 요소들에 따라서 반응을 하면 거기에 따른 적절한 동작을 해줌 절차가 정해져있어서 A다음 B가 아니라 A따로 B따로

event지향 방식. 사용자가 Event에 반응을 하면 거기에 따라 해주는거

CLI Program 방식 -> 절차적인 방식 지금까지 실습에서 한 것들

mainloop는메시지루프 – 사용자로부터 입력, 응답이 들어올 때까지 계속 대기. 무한루프. 윈도우 종료 할 때까지.

#GUI ex

from tkinter import \*

def windowEx():

window=Tk()

window.title('Main Window')

window.geometry('400x100') #가로x세로 widthxheight

window.resizable(width=FALSE,height=FALSE) #윈도우 크기 조절 불가

window.mainloop()

if \_\_name\_\_=='\_\_main\_\_':

windowEx()

메인 루프 전에 모양이나 내용 등 설정 처리가 다 되어있어야함

widget(=control):Window 내용을 구성하기위한 componenet 사용자에게 message 전달 또는 사용자로부터 Event수신

사용자랑 통신하기 위한 역할

사용자랑 통신하거나 수신하기 위한 역할??

윈도우(mainloop() 창)는 표준출력의 영역이 아님. 표준출력 함수를 쓸 수가 없음.

GUI에 출력하려면 윈도우 영역 안에

def windowEx2():

window=Tk()

def eventExitButton():

label3.config(text='click exit button') #버튼 눌리면 label3문장변경

quit() #프로그램 종료

window.title('Widget Test')

window.geometry('400x100') #가로x세로 widthxheight

#window.resizable(width=FALSE,height=FALSE)#윈도우 크기 조절 불가

label1=Label(window,text='Python GUI')

label2=Label(window,text='Python ',font=('Bradley Hand ITC',30),fg='blue') #bradley Hand ITC는 글씨체 fg는 글꼴색 foreground

label3=Label(window,text='GUI',bg='magenta',width=10,height=5,anchor=SE)

exitbutton=Button(window,text='Exit',fg='blue',width=10,height=3,font=('italic',15),command=quit) #command가 콜백callback

label1.pack()

label2.pack()

label3.pack()

exitbutton.pack()

window.mainloop()

if \_\_name\_\_=='\_\_main\_\_':

windowEx2()

label은 child가 되는거

window는 parent

anchor는 정렬방식 동서남북으로씀 +center

callback- 버튼 클릭하면 command 뒤에 함수 호출

버튼은 사용자로부터 선택 받을 때

class Application(Frame): #widow라고 보면 됨. 윈도우 하나

def \_\_init\_\_(self, master=None): #master는 프레임 윈도우? application윈도우에 내부적으로 전달 받는 frame에 대한? Tk객체의 레퍼런스

super().\_\_init\_\_(master)

self.pack()

self.create\_widgets(master) #버튼 생성

def create\_widgets(self,master): #생성자에서 버튼 두개를 직접 생성해줘도 됨

self.hi\_there=Button(self)

self.hi\_there["text"]="Hello Wordld\n(Click me)"

self.hi\_there["command"]=self.say\_hi

self.hi\_there.pack(side="top")

self.quit=Button(self,text="QUIT",fg="red",command=master.destroy)

self.quit.pack(side="bottom")

def say\_hi(self):

print("hi there, everyone!")

def windowEx3():

root=Tk()

root.title('Widget Test')

root.geometry('200x200')

root.resizable(width=FALSE, height=FALSE)

app=Application(master=root)

app.mainloop()

def windowEx4():

root=Tk()

root.title('Widget Test')

root.geometry('400x200')

#root.resizable(width=FALSE, height=FALSE)

button1=Button(root,text='Button1',width=10,height=3)

button2=Button(root,text='Button2',width=10,height=3)

button3=Button(root,text='Button3',width=10,height=3)

button1.pack(side=LEFT)

button2.pack(side=LEFT)

button3.pack(side=LEFT)

#버튼을 세개 연속해서 나란히 배치

btnList=[None]\*3

for i in range(4, 7):

btnList[i-4]=Button(root,text='Buttttttton'+str(i)) #List확인해서 리스트에 담아서 pack() 해도

for btn in btnList:

btn.pack(side=BOTTOM,fill=X,padx=10, pady=10,ipadx=10,ipady=10)

#TOP하면 위에서 아래로 BOTTOM하면 아래에서 위로 생김 BOTTOM - 4가 제일 밑 그리고 6이 제일 위 fill=X는 채우기 하는거 윈도우에서 남은공간 채우기

#pad x, pad y는 버튼(위젯)과 버튼 사이 간격 생김.

#ipad x, ipad y는 위젯 내부() width랑 height는 고정크기 ipad는 상대적으로 간격값 주는거.

#GUI위젯 방식에 에디터가 없기 떄문에 이런식으로 다 맞춰줘야함

root.mainloop()

def windowEx5():

root=Tk()

root.title('Widget Test')

root.geometry('400x200')

def event(ev):

label.config(text=entry.get())

label=Label(root, text='Input Text')

label.pack(side=LEFT)

entry=Entry(root)

entry.pack(side=LEFT)

entry.bind('<Return>',event) #bind - 엔터키가 눌렀을 때 callback

#입력하고 엔터 치면 label 값이 바뀜

#<Button-1> 마우스 왼쪽 버튼 클릭

#<Button-2> 마우스 중간 버튼 클릭

#<Button-3> 마우스 오른쪽 버튼 클릭

#<Double-Button-1> 왼쪽 버튼 더블클릭

#<Return> Enter 키 눌려짐

#<Key> 키가 눌려짐

def func1():

label2.config(text='Radio1')

def func2():

label2.config(text='Radio2')

# def func3():

# if sel==1: #지금은 안됨 sel변수를 꺼내는 과정이 필요한데 알아봐야함

# print('Radio1')

# else sel==2:

# print('Radio2') sel=IntVar()

sel.set(1)

label2=Label(root,text='Select Button')

label2.pack(side=TOP)

rb1=Radiobutton(root,text='Radio1',variable=sel,value=1,command=func1)

rb2=Radiobutton(root,text='Radio2',variable=sel,value=2,command=func2)

rb1.pack(side=TOP)

rb2.pack(side=TOP)

root.mainloop()

----------------------------------------------여담--------------------------------------------------

안드로이드 4가지 구성요소

activity, contact????content? provider, broadcast receiver(메시지 관련), service componenet

sevice에서 rpc또는 다른뭐시기 있음