Postgres-Python Project

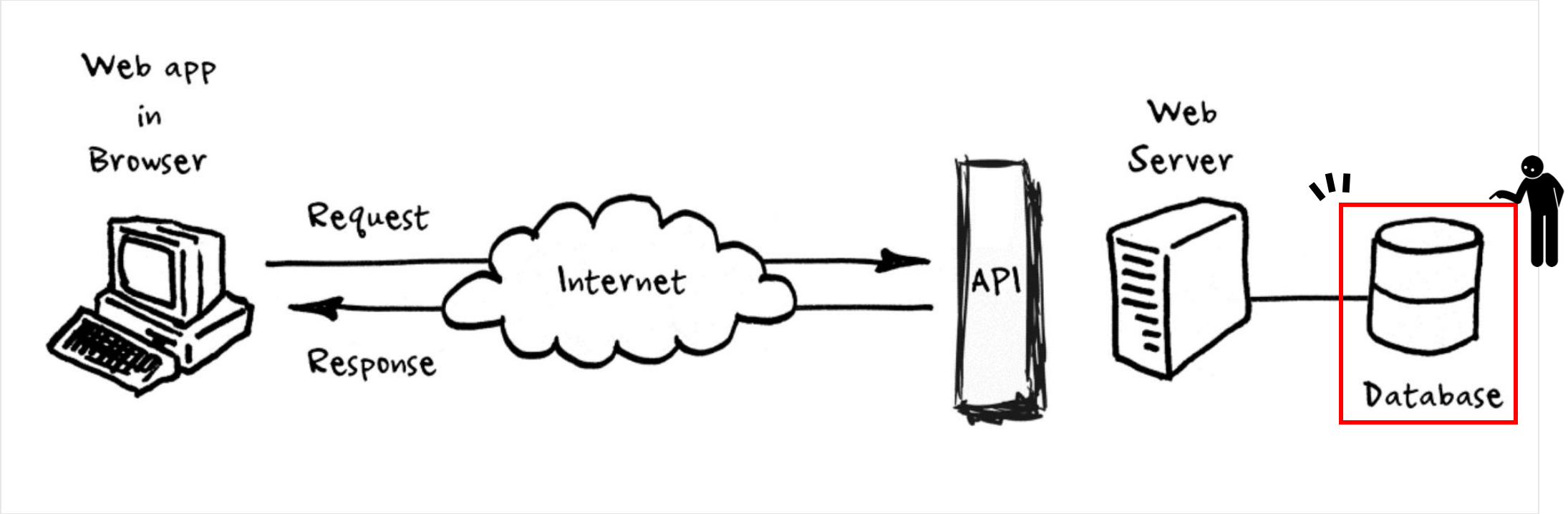
VLDB Lab.

Professor Sangwon Lee

When do we use Database? Common Use Cases

- **Web Applications**: E-commerce platforms, blogs, portal websites.
- **Mobile Applications**: Social networking apps, gaming apps, financial apps.
- **IoT Systems**: Storing data for smart home devices.
- **Enterprise Software**: ERP (Enterprise Resource Planning) and CRM (Customer Relationship Management) systems.

Basic Flow of Web Request - Response

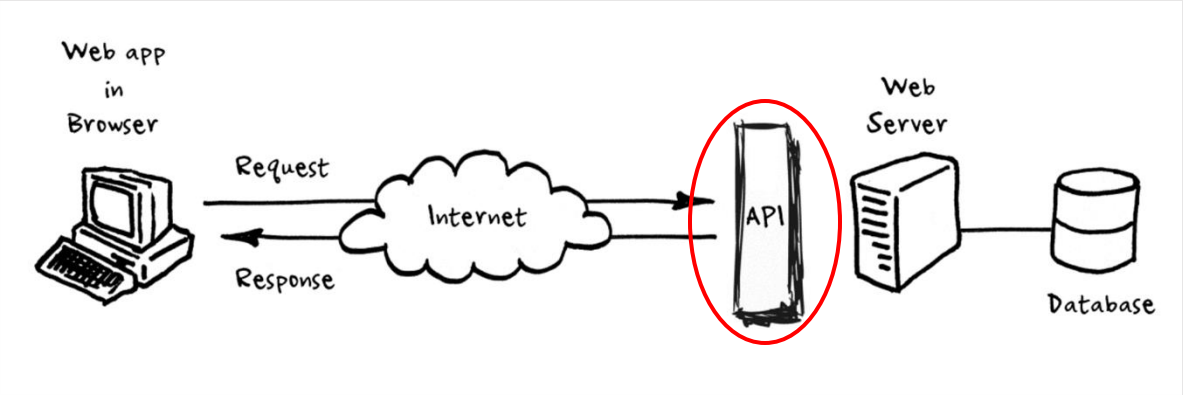


https://www.techfunnel.com/information-technology/application-programming-interface/

Web Basics - API

API

- A **middle intermediary** that facilitates the exchange of data between applications.
- It enables communication between the user interface and the server.



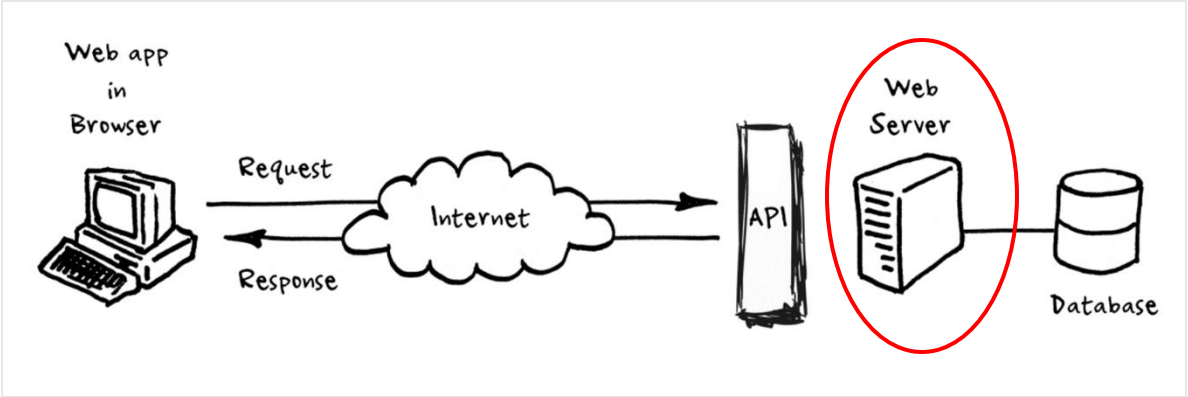
Role of API

- **Communication Bridge:** The API receives requests from the browser or client application via the internet.
- **Abstraction**: The API abstracts the internal complexities of the server and database, exposing only specific endpoints (e.g, /getUser, /updateBook) for the client to interact with.

Web Basics – Web server

Web Server

- Handles and processes requests from the Web App and communicates with the Database to provide the necessary data or functionality.



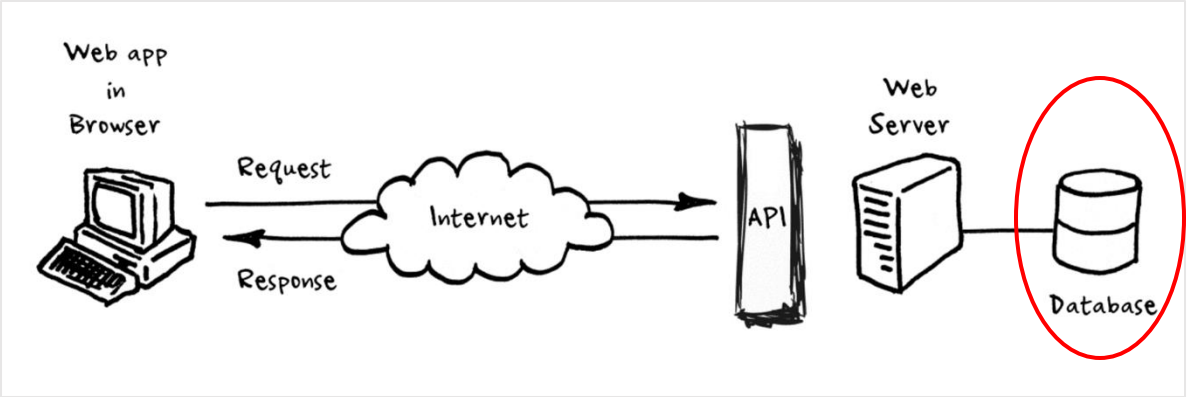
Role of Web Server

- **Request Handling:** Receives request from the browser via API.
- **Communication with the Database:** Sends SQL queries to the database and retrieves the results.
- **Response Generation:** Formats the database results or other processed data into a response that the browser can understand (e.g., JSON, HTML)
- **Security and Authentication**

Web Basics – Database

Database

- Foundation for storing, managing, and retrieving data that the web server uses to fulfill user requests.



Role of Database

- **Data Management**
- **Data Security**
- **Support for Multi-user Access**
- **Data Retrieval**

Popular Languages, Frameworks, and Databases for Web Development

Language & Platform

- Java
  - Spring Framework
- Python ✓
  - Django
  - Flask
- Golang
  - Gin
- JavaScript
  - Node.js
  - Express.js
- …

Database

- Postgres ✓
- MySQL
- Oracle
- MongoDB
- Redis
- Cassandra
- ElasticSearch
- BigQuery
- Neo4j
- …

psycopg2

- To use postgres in Python, you need to install psycopg2.

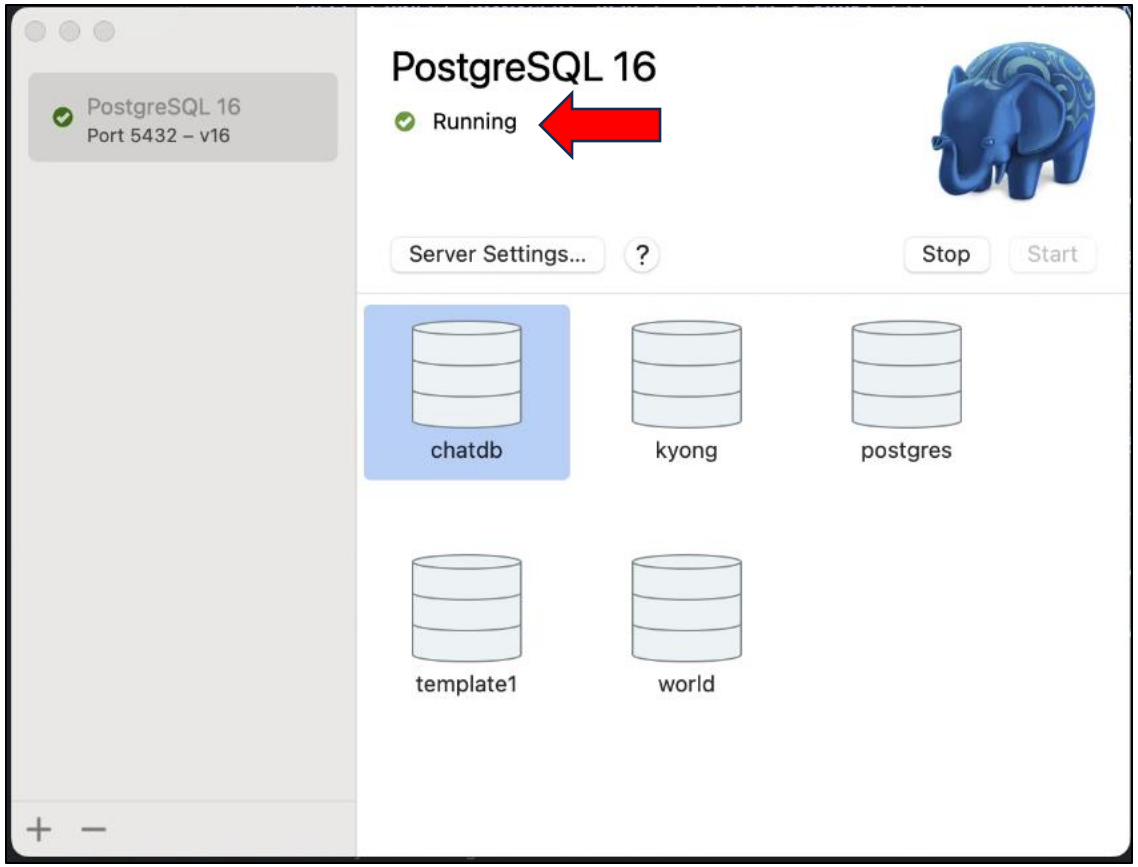
```
$ pip install psycopg2
```

- psycopg2 is a popular PostgreSQL database adapter for Python.

- It allows Python applications to interact with a PostgreSQL database.

Prerequisite

- **The PostgreSQL server must be running before connecting to it in Python.**



Create Table ★

1_create.py

```
import psycopg2

#Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

cursor = conn.cursor()

#Doping EMPLOYEE table if already exists.
cursor.execute("DROP TABLE IF EXISTS scott_EMP")
cursor.execute("DROP TABLE IF EXISTS scott_DEPT")

#Creating table as per requirement
sql ='''CREATE TABLE scott_DEPT
(
    DEPTNO numeric(2) not null CONSTRAINT PK_DEPT PRIMARY KEY,
    DNAME VARCHAR(14) ,
    LOC VARCHAR(13)
);

CREATE TABLE scott_EMP
(
    EMPNO numeric(4) not null CONSTRAINT PK_EMP PRIMARY KEY,
    ENAME VARCHAR(10),
    JOB VARCHAR(9),
    MGR numeric(4),
    HIREDATE DATE,
    SAL numeric(7,2),
    COMM numeric(7,2),
    DEPTNO numeric(2) CONSTRAINT FK_DEPTNO REFERENCES scott_DEPT
);

'''
cursor.execute(sql)
print("Table created successfully........")

conn.commit()
conn.close()
```

Establish Database Connection

Cursor

DROP table if already exists

Execute the query.

Commit the transaction and close the database connection

INSERT ★

2_insert.py

```python
import psycopg2

#Establishing the connection
conn = psycopg2.connect(
   database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

#Setting auto commit false
conn.autocommit = False

#Creating a cursor object using the cursor() method
cursor = conn.cursor()
```

```python
cursor.execute('''
INSERT INTO scott_DEPT VALUES

    (10,'ACCOUNTING','NEW YORK'),

    (20,'RESEARCH','DALLAS'),

    (30,'SALES','CHICAGO'),

    (40,'OPERATIONS','BOSTON');


INSERT INTO scott_EMP VALUES

    (7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20),

    (7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30),

    (7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30),

    (7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20),

    (7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30),

    (7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30),

    (7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10),

    (7788,'SCOTT','ANALYST',7566,to_date('13-7-1987','dd-mm-yyyy')-85,3000,NULL,20),

    (7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10),

    (7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30),

    (7876,'ADAMS','CLERK',7788,to_date('13-7-1987', 'dd-mm-yyyy')-51,1100,NULL,20),

    (7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30),

    (7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20),

    (7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);


''')

print("Records inserted........")

# Commit your changes in the database
conn.commit()
conn.close()
```

SELECT ★

3_select.py

```python
import psycopg2

#Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

#Setting auto commit false
conn.autocommit = False

#Creating a cursor object using the cursor() method
cursor = conn.cursor()

#Retrieving data
cursor.execute('''SELECT * from scott_emp''')

#Fetching 1st row from the table
result = cursor.fetchone();
print(result)

#Fetching all row from the table
result = cursor.fetchall();
print(result)

#join query
cursor.execute('''SELECT * from scott_emp emp, scott_dept dept where emp.deptno = dept.deptno''')
result = cursor.fetchall();
print(result)


conn.commit()
conn.close()
```

SELECT – With Pandas ★

3_1_select_pd.py

```
import psycopg2
import pandas as pd

#Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

#Creating a cursor object using the cursor() method
cursor = conn.cursor()

#join query
cursor.execute('''SELECT * from scott_emp emp, scott_dept dept where emp.deptno = dept.deptno''')
result = cursor.fetchall();
print(result)

colnames = [desc[0] for desc in cursor.description]
df = pd.DataFrame(result, columns=colnames)

print(df)

conn.commit()
conn.close()
```

Import pandas

Manipulate Data using Pandas

UPDATE ★

4_update_exception.py

```python
import psycopg2

#Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

#Setting auto commit false
conn.autocommit = False

#Creating a cursor object using the cursor() method
cursor = conn.cursor()

# Preparing the query to update the records
sql = '''UPDATE scott_emp SET sal = sal + 1000 WHERE job = 'PRESIDENT' '''
try:
    # Execute the SQL command
    cursor.execute(sql)

    # Commit your changes in the database
    conn.commit()
except:
    # Rollback in case there is any error
    print("Roll back!")
    conn.rollback()

# Closing the connection
conn.commit()
conn.close()
```

Rollback the transaction

DELETE ★

5_delete_template.py

```python
import psycopg2

#Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port= '5432'
)

#Setting auto commit false
conn.autocommit = False

#Creating a cursor object using the cursor() method
cursor = conn.cursor()

# Preparing the query to delete records
sql = "DELETE FROM scott_emp WHERE job = '%s'" % ('PRESIDENT')

try:
    # Execute the SQL command
    cursor.execute(sql)

    # Commit your changes in the database
    conn.commit()
except:
    print("Roll back!")
    # Roll back in case there is any error
    conn.rollback()

# Closing the connection
conn.commit()
conn.close()
```

Rollback the transaction

DROP ★

6_drop.py

```
import psycopg2

# Establishing the connection
conn = psycopg2.connect(
    database="postgres", user='postgres', password='postgres', host='localhost', port='5432'
)

# Setting auto commit false
conn.autocommit = False

# Creating a cursor object using the cursor() method
cursor = conn.cursor()

cursor.execute("DROP TABLE scott_emp")
cursor.execute("DROP TABLE scott_dept")

print("Table dropped... ")

# Closing the connection
conn.commit() # 안하면 자동 롤백
conn.close()
```