LECTURE 5
INTRODUCTION TO ENTERPRISE SYSTEM

LEK HSIANG HUI

LEARNING OBJECTIVES

At the end of this lecture, you should understand:

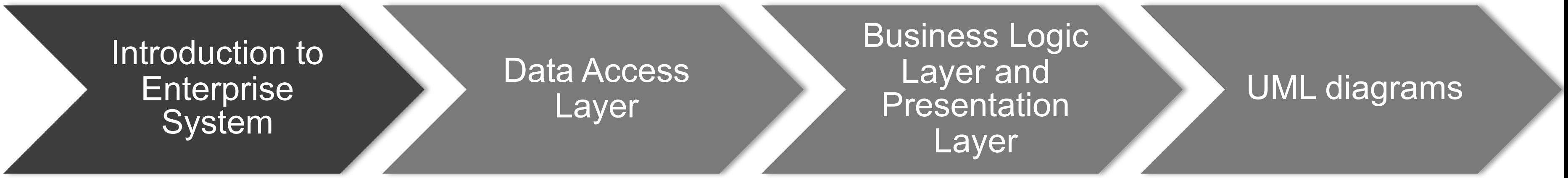
- The basics and design of enterprise systems
- The 3 major layers of enterprise systems: data access layer, business logic layer, and presentation layer
- Overview of the most important UML diagram used in the SDLC

BEFORE WE START...

Please note that the 2 major players for Enterprise System Development Framework in the industry is Java Enterprise Edition (JavaEE) and .NET

- They are a lot more established and have a more elaborate structure to support Enterprise System Development
- For this course (since IT5001 was in Python), the code examples given in this lectures will be in Python as far as possible
- But good to understand how such framework works (e.g. JavaEE)

INTRODUCTION TO ENTERPRISE SYSTEM


Introduction to Enterprise System
Data Access Layer
Business Logic Layer and Presentation Layer
UML diagrams

QUESTION TIME

1. After 4 weeks of classes, what are some characteristics of Enterprise system?

ENTERPRISE SYSTEMS

To design and develop such systems requires much planning and resources

- Important to understand that systems development is not just about coding
- But involves a lot of requirements gathering and analysis
- Team working on a project rather than 1 or 2 people
- Need to have a way to "divide and conquer" (divide problem into small tasks)

ENTERPRISE SYSTEMS

The system design and development process needs to be scalable and well-separated

- Different people working on different things (e.g. analysts designing architecture of system, developers doing coding, designers designing UI, etc)

The design must also be able to accommodate potential future enhancements

- Systematic design
- Solid fundamental architecture

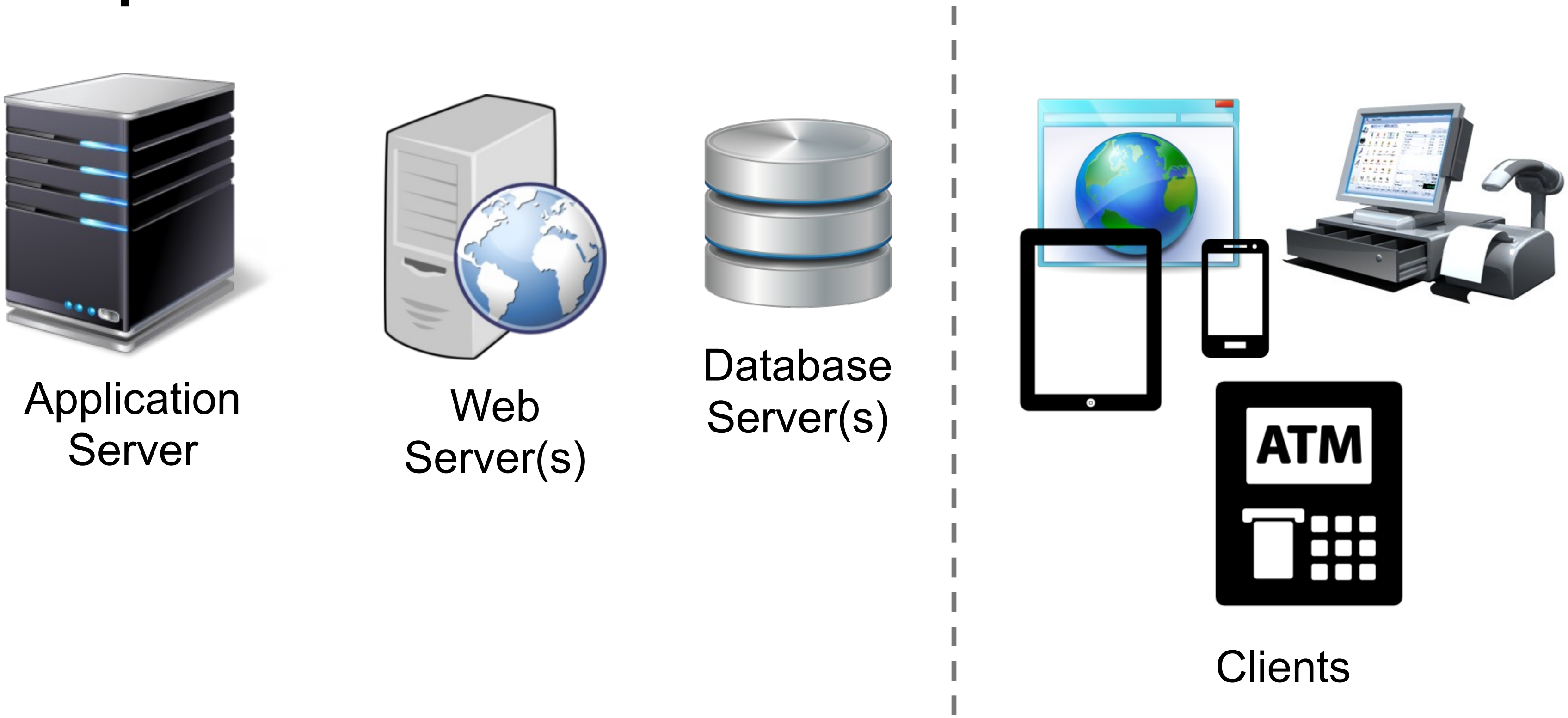
ENTERPRISE SYSTEMS

Enterprise systems requirements:

- Able to support multiple (concurrent) users
- Able to interface with other systems (e.g. payment systems)
- Able to support multiple ways to access the service (e.g. browser, smartphone, hardware, etc)
- Robust and scalable
- Able to persist the data even if the application crashes

GENERAL ENTERPRISE SYSTEMS COMPONENTS

Generally, enterprise systems consists of the following components:



Application Server

Web Server(s)

Database Server(s)

Clients

SERVER

The “server” section is divided into multiple servers each in charge of different aspects of the system

Database Server(s): persistent storage of data

Application Server: provides the business logic (connects to DB Server)

Web Server(s): consumes the services from the application server to fulfil client requests coming from the web

- Some application server comes with web server capability, so they are combined

DATABASE SERVER

There are different types of databases:

- Relational Database (RDBMS)
- NoSQL

RDBMS is often/traditionally used for enterprise system

Each DB Server can consist of multiple databases

- Each database can have multiple tables
  - Each table has a schema which describes the structure of the data
  - Each record is represented as a row, and its fields are represented as columns

DATABASE

Person Table

id	name	age	contact_num	…
1	John	20	91234567	…
2	Mary	18	97654321	…
…				

Need to have a primary key field that uniquely identifies each record

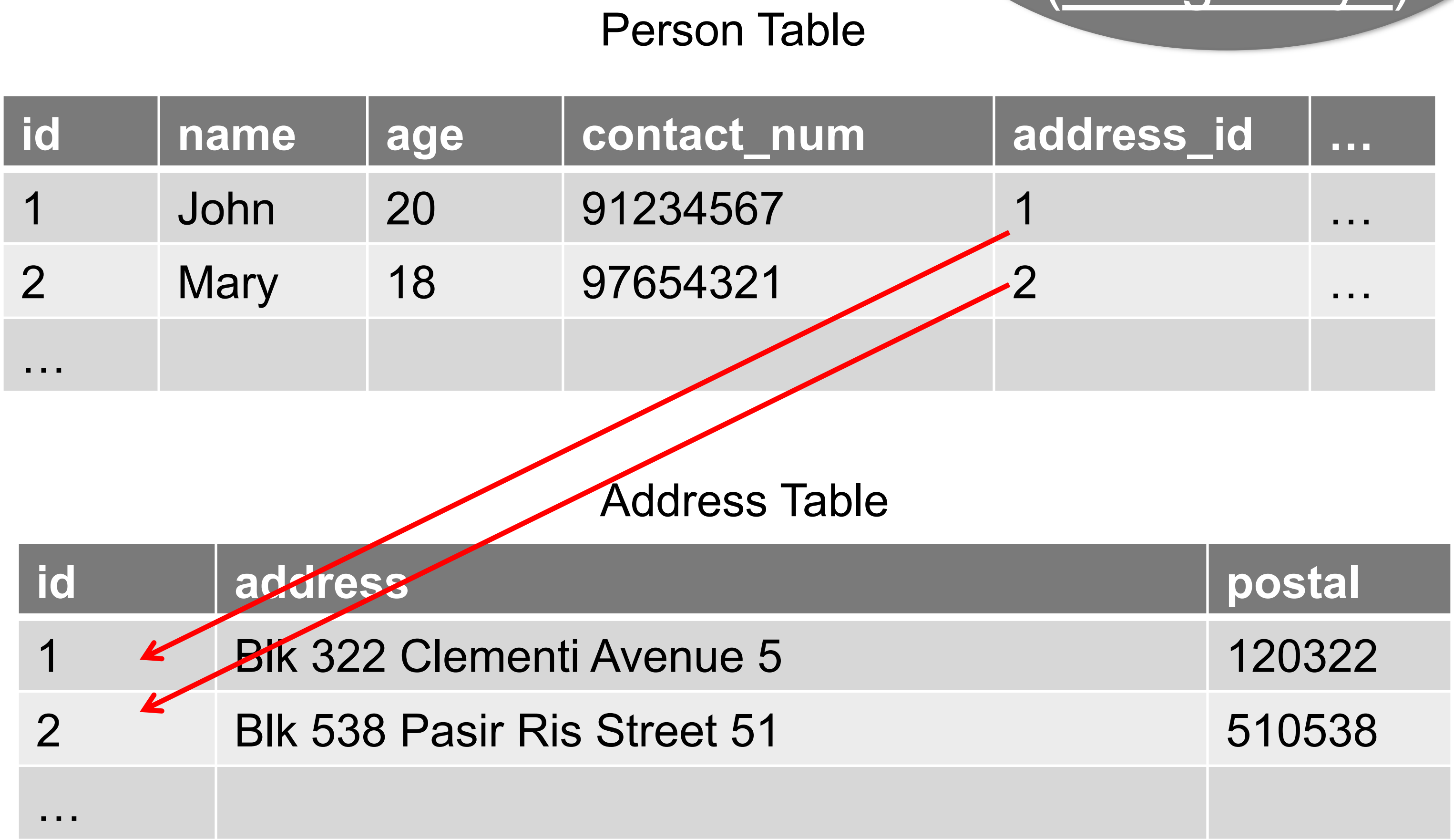
Person Table Schema

Person

id: int (primary_key, auto_increment)
name: string
age: int
contact_num: string
…

DATABASE

Tables can be linked to each other (foreign keys)



Person Table

id	name	age	contact_num	address_id	…
1	John	20	91234567	1	…
2	Mary	18	97654321	2	…
…					

Address Table

id	address	postal
1	Blk 322 Clementi Avenue 5	120322
2	Blk 538 Pasir Ris Street 51	510538
…		

APPLICATION SERVER

Contains the business logic

- Algorithms/codes to address a business task
- E.g. Capture the list of staff
- Prevent access to the system by using login mechanism
- etc

Connects to DB Server:

- To access data
- To insert/update/delete data

WEB SERVER

If clients access the services through the web, there should be a web server

- Takes in request from the user (e.g. web browser)
- Connects to the application server and call the relevant services
- Application server returns the results to the web server
- Web server displays the result in a properly formatted manner back to the user

CLIENT

The client provides the interface to use the services provided by the server

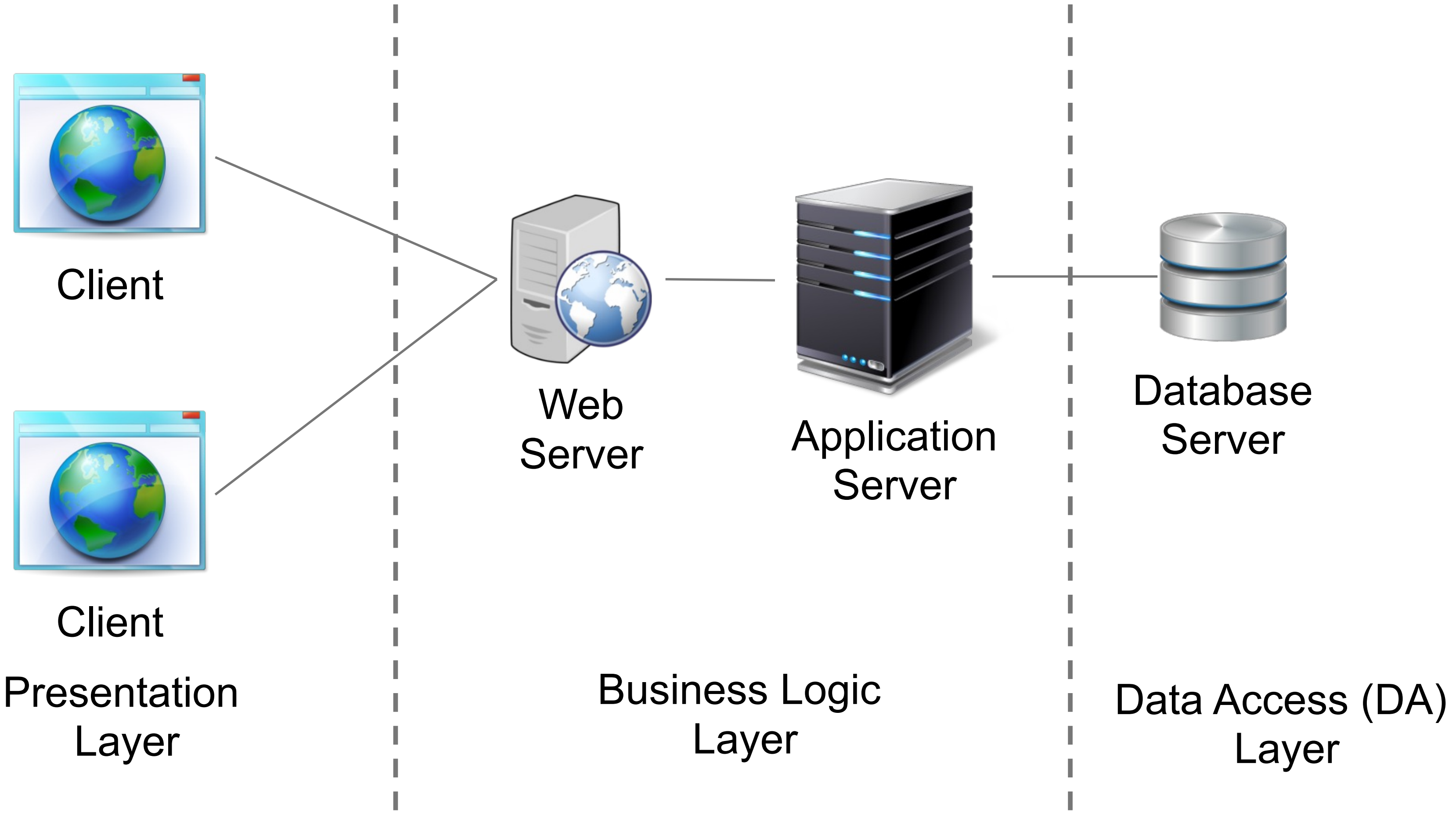
It can talk directly to the application server or the web server

- Web browser talks to the web server
- ATM connects to an application server securely to process transactions

Servers can also become a “client” to another server

- E-commerce system connects to PayPal server to verify credit card details

MULTI-TIER / MULTI-LAYER ARCHITECTURE



Client

Client

Web Server

Application Server

Database Server

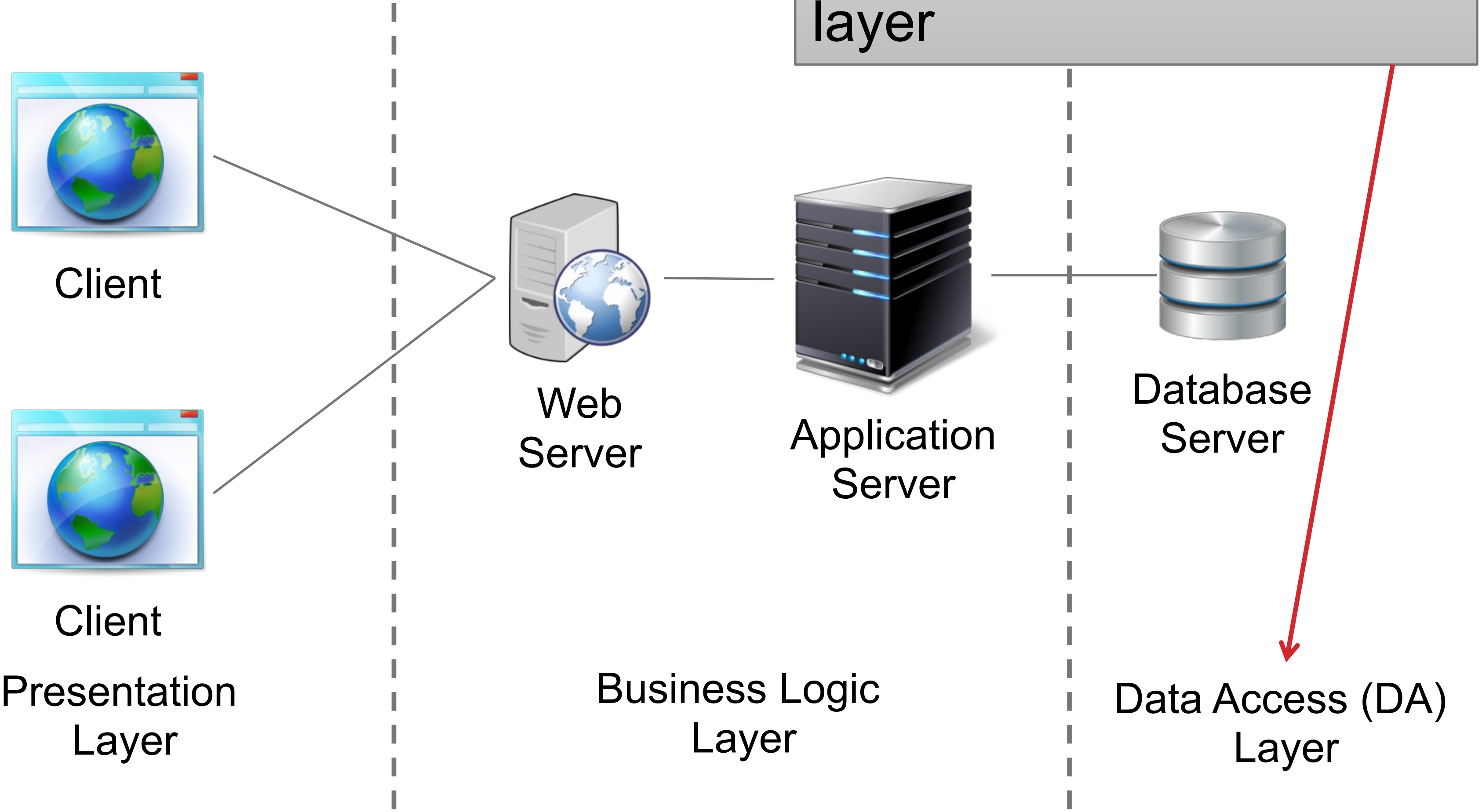
Presentation Layer

Business Logic Layer

Data Access (DA) Layer

MULTI-TIER / MULTI-LAYER ARCHITECTURE

Note that the application has a systematic way to access the database using the data access layer



DATA ACCESS LAYER

Introduction to Enterprise System
Data Access Layer
Business Logic Layer and Presentation Layer
UML diagrams

DA LAYER

```
…
# Connect to db
conn = sqlite3.connect(…)

cursor = db.cursor()

# Execute SQL select statement
cursor.execute("SELECT * FROM person")

#list of person
persons = []

# Get and display one row at a time
rows = cursor.fetchall()
for row in rows:
    person = Person(row[0], row[1], ...)
    persons.append(person)

# Close the connection
db.close()
```

Most of the operations would need DB access - Imagine the amount of repeated codes!

✗

DA LAYER

Furthermore, imagine the amount of conversion of database records to objects!

```
…
# Connect to db
conn = sqlite3.connect(…)

cursor = db.cursor()

# Execute SQL select statement
cursor.execute("SELECT * FROM person")

#list of person
persons = []

# Get and display one row at a time
rows = cursor.fetchall()
for row in rows:
    person = Person(row[0], row[1], ...)
    persons.append(person)

# Close the connection
db.close()
```

DA LAYER

If we had to change the database connection configuration later on, it will be a disaster!

```
…
# Connect to db
conn = sqlite3.connect(…)

cursor = db.cursor()

# Execute SQL select statement
cursor.execute("SELECT * FROM person")

#list of person
persons = []

# Get and display one row at a time
rows = cursor.fetchall()
for row in rows:
    person = Person(row[0], row[1], ...)
    persons.append(person)

# Close the connection
db.close()
```

SIMPLE DATABASE DEMO

1. If you want to follow along, download L5_sample_codes.zip

DA LAYER

Systematic:

- The application (different methods) does not just issue database connect commands and SQL statements each time when there is a need to access the DB
- Instead, 1 or more Data Access (DA) classes are created for interacting with the DB
- Any methods that require DB access goes through these DA classes

DA LAYER

The use of Object-Relational Mapping (ORM) frameworks is preferred

- Convert data from DB to objects and objects to DB records automatically
- Programmers only need to deal with objects
- No more concept of foreign keys, join tables, VARCHAR(...), SQL statements, MySQL, SQLServer, etc

DA LAYER EXAMPLE (DJANGO)

```
from django.db import models


# define the Person model
class Person(models.Model):
    first = models.CharField(max_length=60)
    last = models.CharField(max_length=60)


# create a person object
p = Person(first="John", last="Doe")

# call the save() method to save the person to db
p.save()
```

DA LAYER EXAMPLE (JAVAEE)

```
// DA class
public class DAHelper{
  EntityManager em;
  …
  public List<Person> searchPerson(String name){
    Query q = em.createQuery("SELECT p FROM PERSON p 
where p.name = :val");
    q.setParameter("val", name);
    return q.getResultList();
  }
}
```

```
DAHelper daHelper = …;

// search for person by name
// return a list of person
public ArrayList<Person> search(String name){
  return daHelper.searchPerson(name);
}
```

DA LAYER EXAMPLE

All database access (insert, search, delete, update) are handled by the framework (DA layer)

- Conversion of objects to database records
- Retrieval of database records to objects
- Developers for these classes only need to focus on the business logic and assume that the DA layer does its job

Achieve better separation of concerns

ORM EXAMPLE

id	name	age	contact_num	address_id	…
1	John	20	91234567	1	…
…					

id	address	postal
1	Blk 322 Clementi Avenue 5	120322
…		

```
class Person {
  private long id;
  private String name;
  private int age;
  private String contactNumber;
  private Address address;
  …
}
```

```
class Address{
  private long id;
  private String address;
  private String postal;
}
```

ORM EXAMPLE

id	name	age	contact_num	address_id	…
1	John	20	91234567	1	…
…					

person_id	address_id
1	1
…	

id	address	postal
1	Blk 322 Clementi Avenue 5	120322
…		

```
class Person {
  private long id;
  private String name;
  private int age;
  private String contactNumber;
  private Address[] addresses;
  …
}
```

```
class Address{
  private long id;
  private String address;
  private String postal;
}
```

BUSINESS LOGIC LAYER AND PRESENTATION LAYER

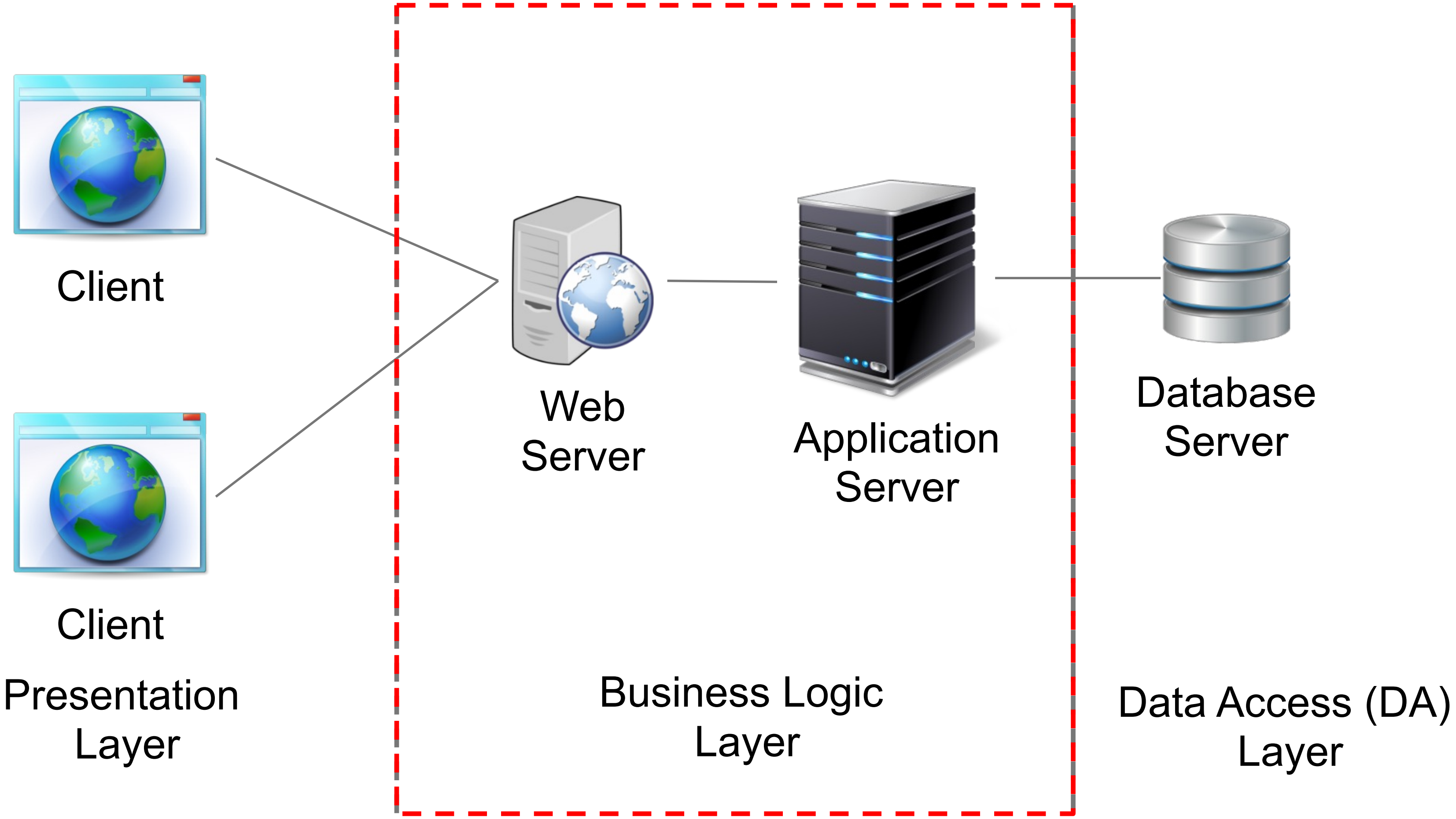
Introduction to Enterprise System

Data Access Layer

Business Logic Layer and Presentation Layer

UML diagrams

MULTI-TIER ARCHITECTURE



BUSINESS LOGIC LAYER

Receives request from the client

Fetches relevant data using the DA layer

Does computation on the data

Sends data to the presentation layer

PRESENTATION LAYER

Receives the raw data from business logic layer

Presents the data in a readable format

Provides user interface:

- User interacts with the system through the presentation layer
- Any request to do something is then directed to the business logic layer
- It often does not much logic beyond data validation

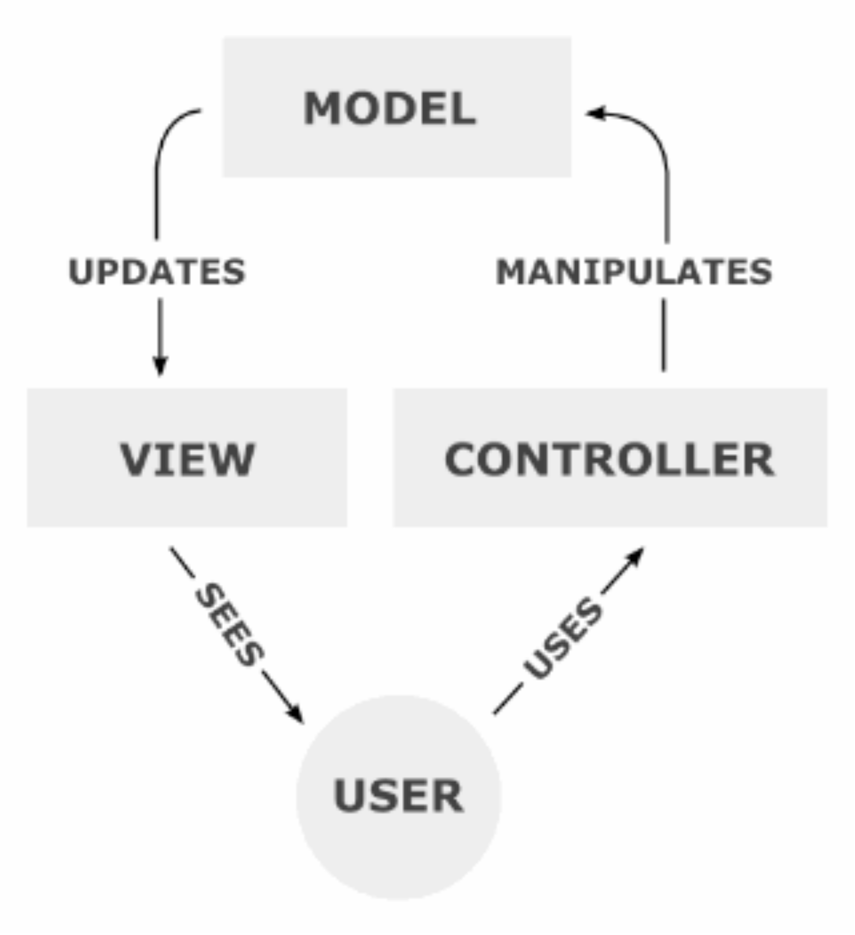
DESIGN PATTERNS

There are various ways to design the business logic layer and presentation layer

Software design patterns are general solution to a common problem

- Problem: how to have a clean design which separates the requests logic, display logic, business logic
- Web-based applications often use the Model-View-Controller design pattern

MODEL/VIEW/CONTROLLER (MVC)



Model : Domain object (updates the view)
View : Presentation
Controller : receives commands from user and invoke the model's methods (direct traffic)

One of the most popular design pattern nowadays, commonly used for web applications

UML DIAGRAMS

Introduction to Enterprise System

Data Access Layer

Business Logic Layer and Presentation Layer

UML diagrams

UML

Apart from activity diagram and use case diagram, UML also defines other diagrams useful for SDLC

2 main categories:

- Structural diagrams
    - Class diagram
    - Package diagram
- Behavioral diagrams
    - Activity diagram
    - Use case diagram
    - State Machine diagram / statechart
    - Sequence diagram

Here are the list of the most useful diagrams

SUMMARY

Enterprise systems follows a server/client architecture

When designing the systems, it is important to “divide conquer” and try to achieve separation of concerns

For example, database access should be separated into a tier/layer

Likewise for business logic and presentation logic

WHAT'S NEXT?

Data Modeling