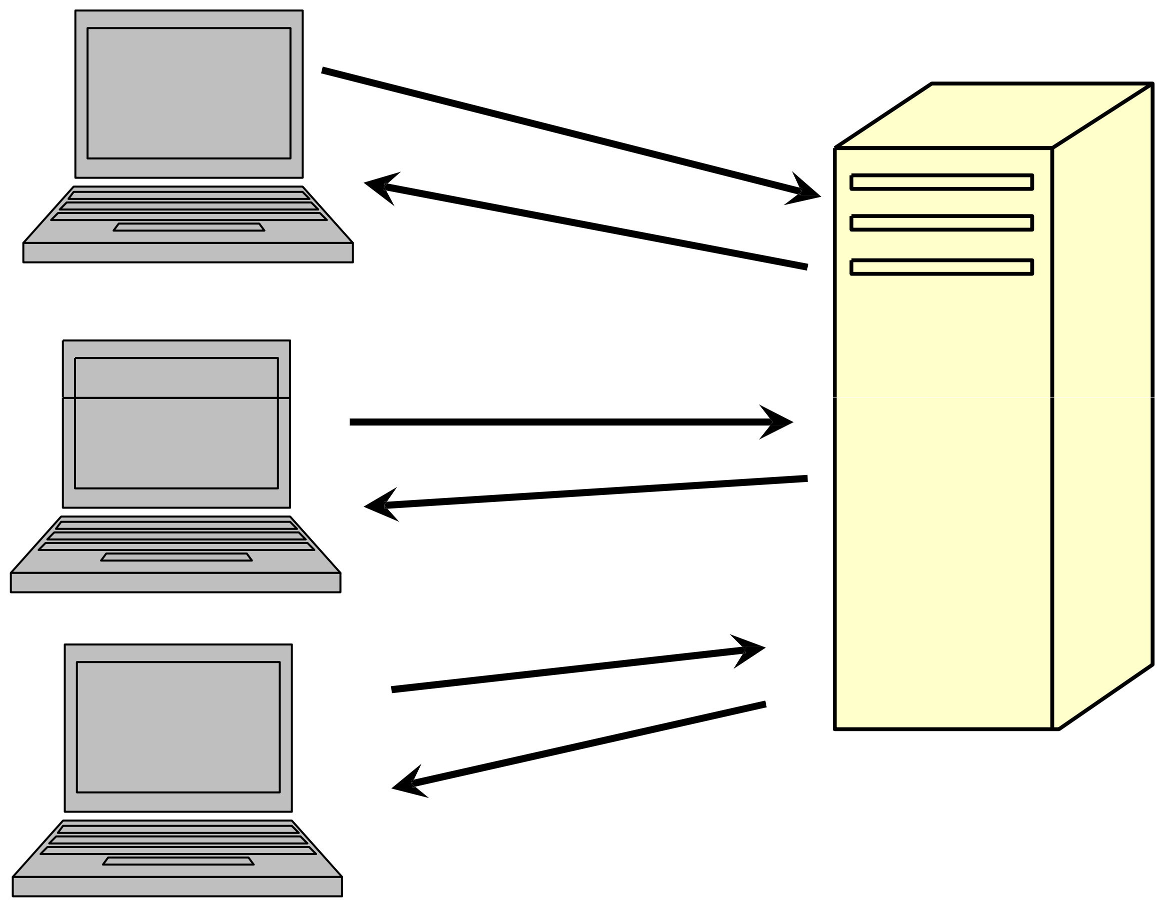
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# Web Development Models

## Content

- Web application architecture: client-server
- 3-layer architecture and MVC model

## Client-Server Model

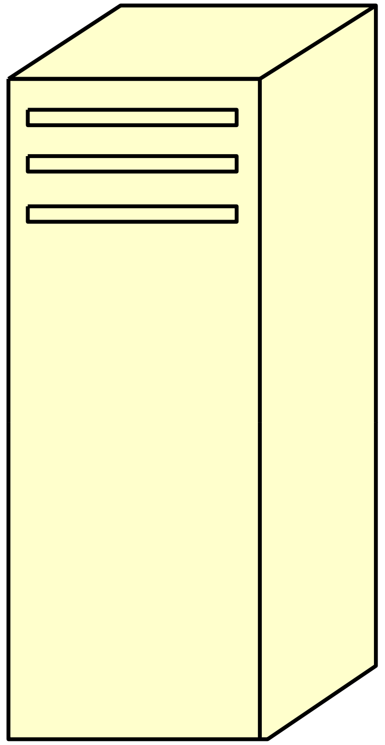


Client side

Server side

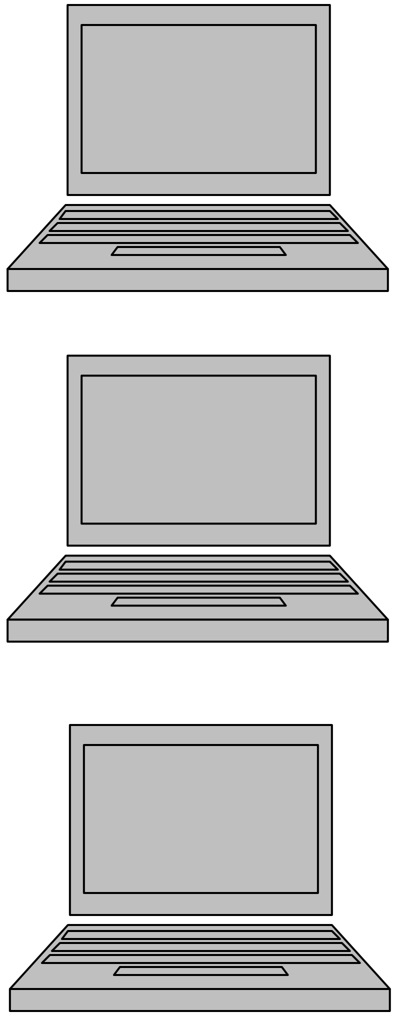
## Server Roles

- Manage and store data, including:
  - User data
  - Application data
- Provide processing services for data
- Centralize data
- Manage user authentication, authorization mechanisms via login function



# Client Roles

- Provide user interface
- Can store some small data (using cookie)
- Can process data (check validity of data that are entered by users
  - Thin client: only provides user interface, centralize data processing on server side
  - Thick client: realizes data processing on client side
- Can be accessed from everywhere with minimal software installation

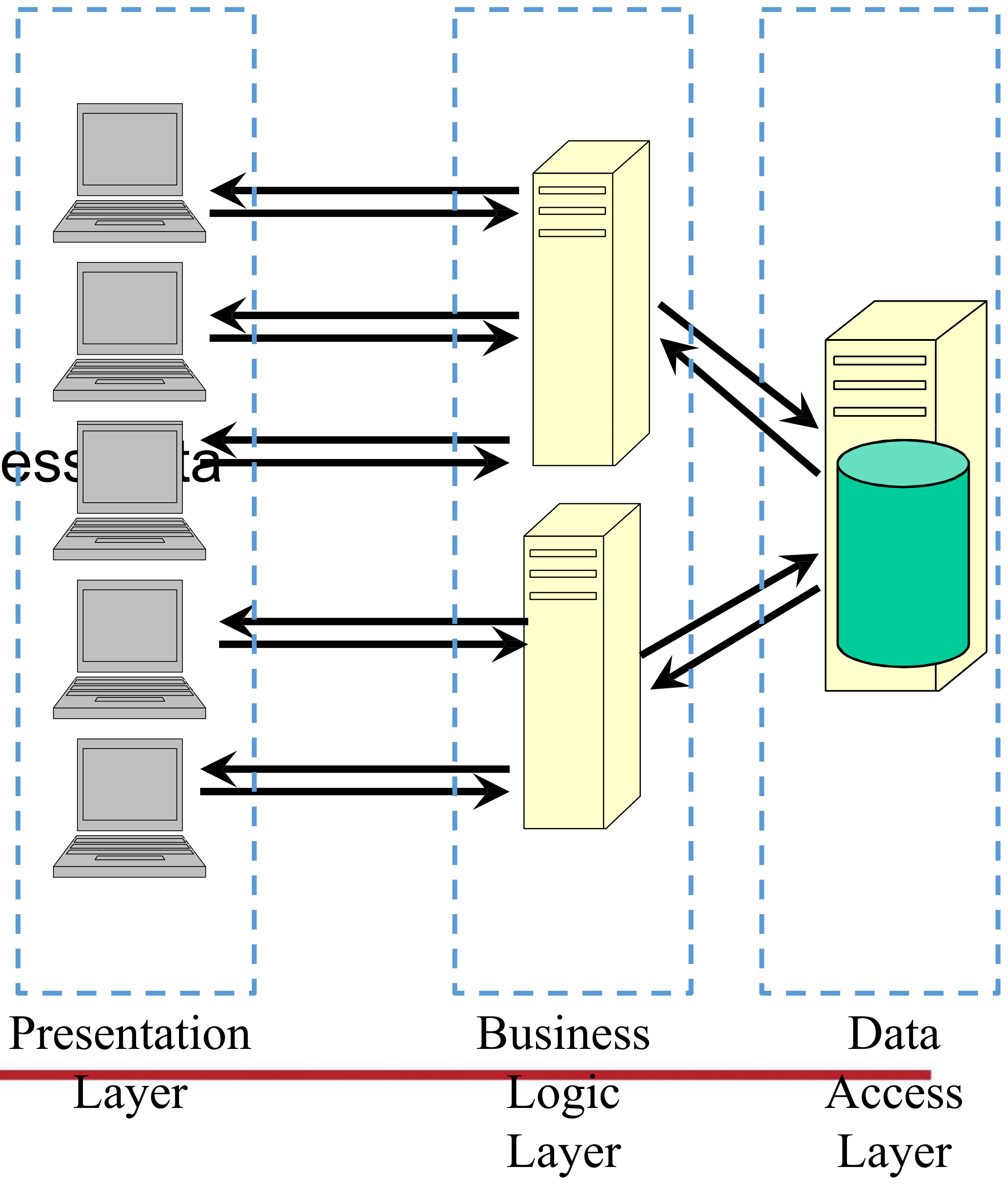


## Client-Server Advantages

- Centralized storage and processing.
- No data redundancy
- Enhance the ability of sharing data
  - If data are distributed on multi-systems of users, it will cause difficulties in sharing the data  because each system has its own database architecture

# 3-Layer Architecture

- Presentation Layer
  - Provides interface and processing
- Business Logic Layer
  - Manages application connections and process ta
- Data Access Layer
  - Stores and accesses data in low-level



## 3-Layer Architecture Advantages

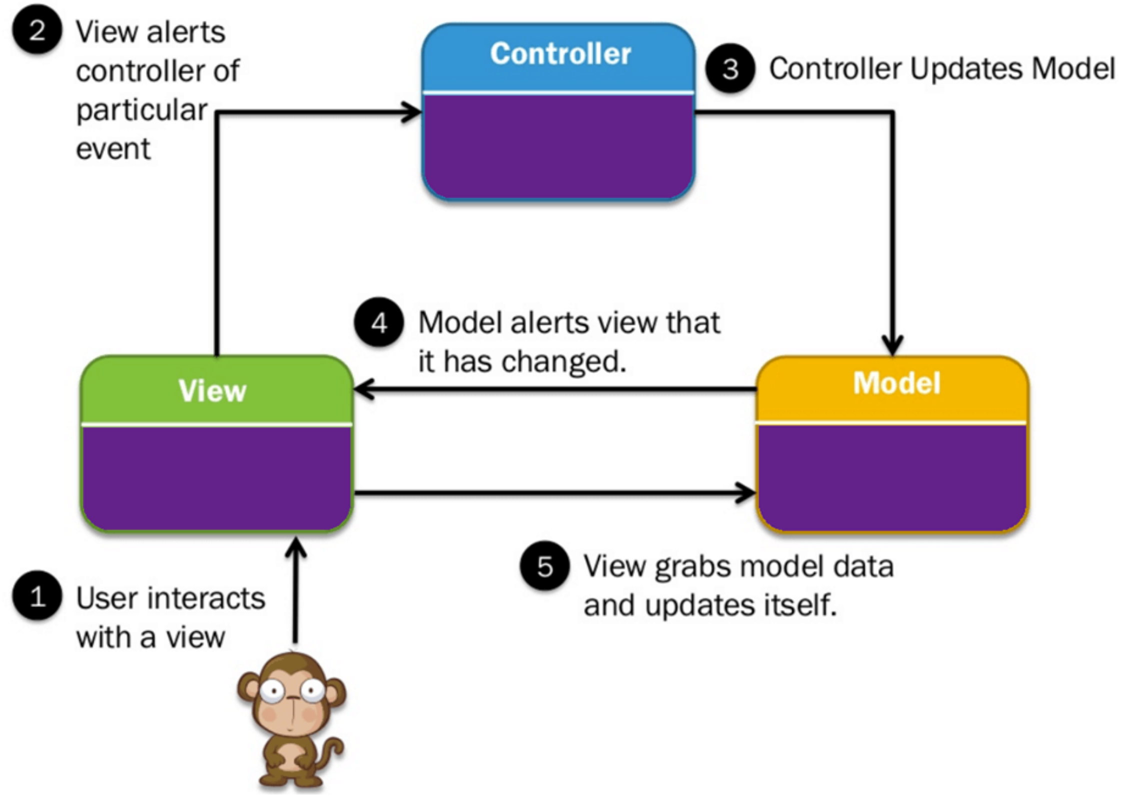
- Centralized Database can be accessed by many servers at the same time
- Allow load balance of user connections on many application servers
- **Data Access Layer** is consistently designed with hardware in order to serve specific its tasks:
  - Data manipulations: update, insert, remove, etc
  - Need more reliable hard drives
- **Business Logic Layer** are designed to provide connection points for user connections and run multi-applications
  - Need more computing power of CPU

## Programming Languages

| Client | Server | Database |
|---|---|---|
| Html | PHP | SQL |
| JavaScript | JavaScript | NoSQL |
| CSS | Java, JSP | |
| | Python | |

# MVC Development Model

- Architectural Pattern from Smalltalk (1979)
- Decouples data and presentation
- Eases the development


2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
1 User interacts with a view
5 View grabs model data and updates itself.

## MVC – The Model

- The “Model” contains the data
- Has methods to access and possibly update it’s contents.
- Often, it implements an interface which defines the allowed model interactions.
- Implementing an interface enables models to be pulled out and replaced without programming changes.

## MVC – The View

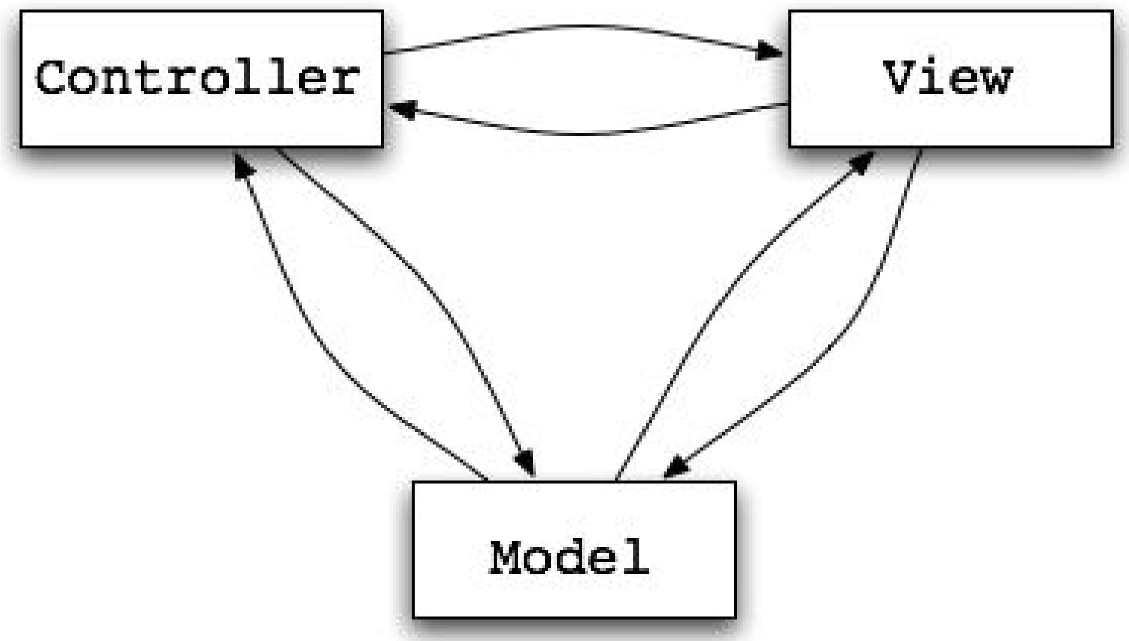
- The View provides a visual representation of the model.
- There can be multiple views displaying the model at any one time.
  - For example, a companies finances over time could be represented as a table and a graph.
  - These are just two different views of the same data.
- When the model is updated, all Views are informed and given a chance to update themselves.

## MVC – The Controller

- It interprets mouse movement, clicks, keystrokes, etc
- Communicates those activities to the model – eg: delete row, insert row, etc

## Example Control Flow in MVC

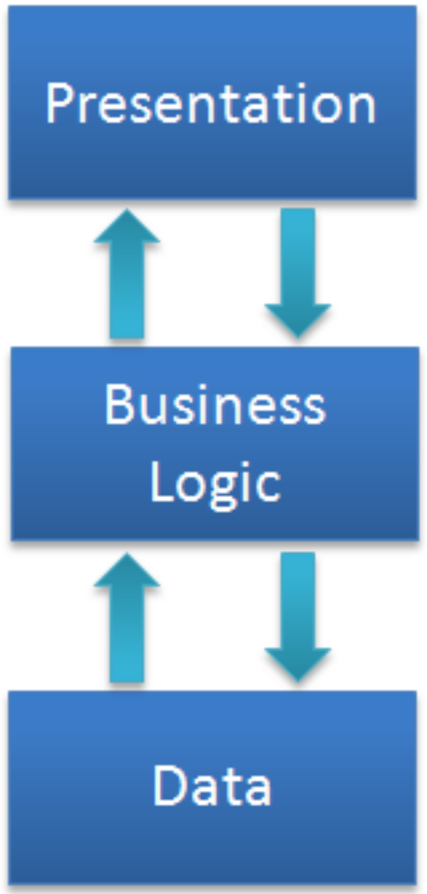
- User interacts with the VIEW UI
- CONTROLLER handles the user input (often a callback function attached to UI elements)
- CONTROLLER updates the MODEL
- VIEW uses MODEL to generate new
- UI waits for user interaction

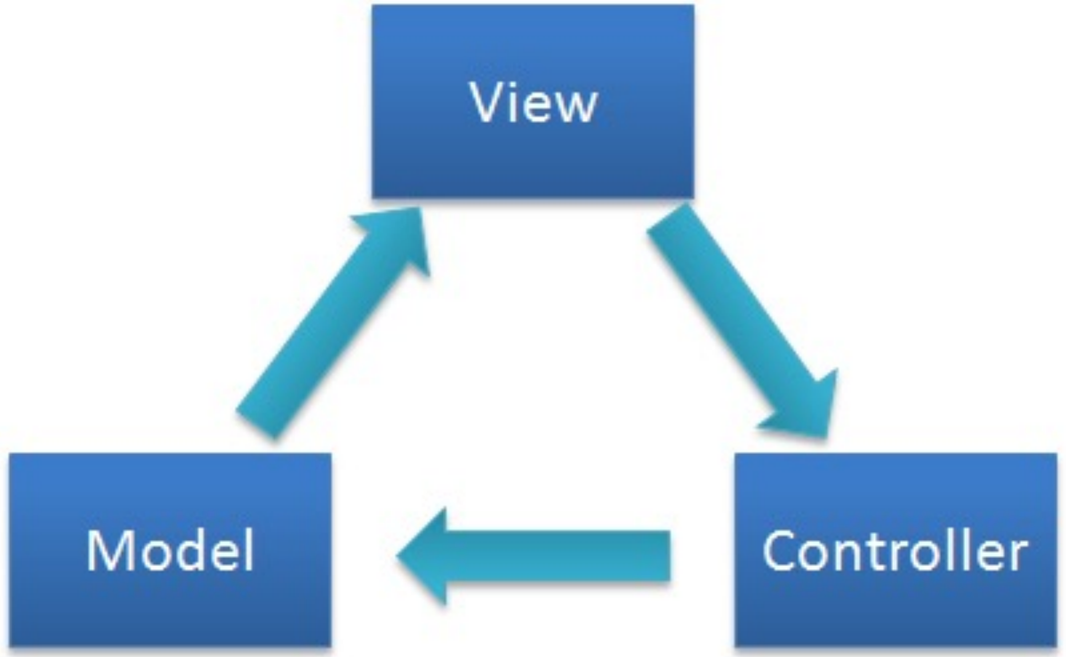


## MVC Advantages

- MVC decouples the model, view, and controller from each other to increase flexibility and reuse.
  - You can attach multiple views to the model without rewriting it.
  - You can change the way a view responds to user input without changing the visual presentation. For example, you might use a pop-up menu instead of keyboard command keys.

## 3-Layer vs. MVC


Presentation
Business Logic
Data


View
Model
Controller