**Bài 1: Cơ sở dữ liệu quan hệ**

* Cơ sở dữ liệu : database, là nhóm các dữ liệu có tổ chức.
* Mô hình dữ liệu: cơ chế lưu trữ, quản lý và truy vấn dữ liệu.

1. Cơ sở dữ liệu quan hệ:

Là cơ sở dữ liệu được xây dựng trên mô hình dữ liệu quan hệ.

Mô hình dữ liệu quan hệ: Dữ liệu trong cơ sở dữ liệu quan hệ được tổ chức thành các table, column (trường, attribute, field) và row (tuples, record). Mỗi table đại diện cho 1 entity (thực thể) tương ứng với 1 class trong Java Bean.

1. Hệ quản trị CSDL: DBMS (Database Management System) là một ứng dụng/phần mềm máy tính được sử dụng để quản lý cơ sở dữ liệu.

DBMS có chức năng:

* Định nghĩa dữ liệu.
* Cập nhật dữ liệu.
* Truy vấn dữ liệu.
* Quản lý dữ liệu (clone, backup, share, phân quyền, bảo mật…)

1. MySQL là hệ quản trị CSDL quan hệ có mã nguồn mở.

MySQL gồm có 2 thành phần: MySQL Server là máy chủ dịch quản trị CSDL; MySQL Client là máy khách cho phép tương tác với máy chủ MySQL server.

1. SQL (Structured Query Language) là ngôn ngữ truy vấn có cấu trúc, được sử dụng để thao tác với cơ sở dữ liệu quan hệ.

SQL có nhiều phiên bản, tùy thuộc vào hệ quản trị cơ sở dữ liệu (MySQL, SQL Server, MS SQL) mà ngôn ngữ SQL có các câu lệnh với cú pháp khác nhau.

SQL có 2 nhóm câu lệnh chính là DML và DDL

🡪DML (Data Manipulation Language) là ngôn ngữ thao tác dữ liệu, gồm các câu lệnh **Insert, Delete, Update, Select**.

🡪DDL (Data Definition Language) là ngôn ngữ định nghĩa dữ liệu, gồm các câu lệnh

**Create, Alter, Drop, Rename**.

🡪DCL (Data Control Language) là ngôn ngữ điều khiển dữ liệu, có chức năng Grant (phân quyền) và Revoke (thu hồi quyền).

1. Sự khác nhau của Delete, Truncate và Drop

🡪Delete: xóa một hay tất cả các bản ghi trong bảng theo một điều kiện nào đó, dữ liệu có thể phục hồi được.

🡪Truncate: xóa tất cả các bản ghi trong bảng, giải phóng bộ nhớ và không thể phục hồi được (không thể rollback được).

🡪Drop: Xóa cơ sở dữ liệu hoặc bảng khỏi cơ sở dữ liệu. Dữ liệu không thể được phục hồi (không thể rollback được).

**Lưu ý**: nên cẩn thận khi xóa dữ liệu bằng Truncate và Drop vì dữ liệu không thể được phục hồi.

**Bài 2: Thiết kế và tạo CSDL**

1.Constraint: các ràng buộc, quy tắc được quy định cho các bảng nhằm hạn chế dữ liệu đưa vào bảng 🡪 Dữ liệu chính xác và đáng tin cậy.

Constraint áp dụng cho các column hay toàn table.

🡪NOT NULL: không cho phép giá trị null.

🡪UNIQUE: mỗi giá trị trong cột là duy nhất.

🡪PRIMARY KEY: khóa chính = NOT NULL + UNIQUE.

🡪FOREIGN KEY: khóa ngoại cho bảng, liên kết bảng hiện tại đến một bảng khác.

🡪CHECK: kiểm tra điều kiện đầu vào .

🡪DEFAULT: thiết lập giá trị mặc định cho cột.

2. Các bước chuẩn hóa dữ liệu (3NF)

a. Bước chuẩn hóa 1NF:

🡪Loại bỏ các thuộc tính đa trị.

🡪Loại bỏ các thuộc tính được tính toán từ các thuộc tính khác.

b. Chuẩn hóa 2NF:

🡪 Đạt chuẩn hóa 1NF (tức là dữ liệu đã được loại bỏ các thuộc tính đa trị và loại bỏ các thuộc tính được tính toán từ các thuộc tính khác).

🡪Loại bỏ các phụ thuộc bộ phận, tức là loại bỏ các thuộc tính không khóa phụ thuộc vào một bộ phận nào đó của khóa chính – primary key.

c. Chuẩn hóa 3NF:

🡪Đạt chuẩn hóa 2NF(tức là dữ liệu đã được loại bỏ các thuộc tính đa trị, dữ liệu đã được loại bỏ các thuộc tính được tính toán từ các thuộc tính khác; loại bỏ các phụ thuộc bộ phận – các thuộc tính không khóa phụ thuộc vào một bộ phận nào đó của khóa chính.)

🡪Loại bỏ các phụ thuộc bắc cầu.

3. Phân tích và thiết kế CSDL:

(1) Xác định mục đích của CSDL.

(2) Tìm hiểu, tổ chức thông tin cần lưu trữ.

(3) Phân chia thông tin vào các bảng.

(4) Xác định các trường/field/thuộc tính của các bảng.

(5) Xác định các Primary Key của các bảng.

(6) Xác định các Foreign Key của bảng.

(7) Làm mịn thiết kế: Phân ra mối quan hệ n-n thành các mối quan hệ 1-n, 1-1.

(8) Áp dụng các nguyên tắc chuẩn hóa dữ liệu (1NF/2NF/3NF).

**Bài 3: Thao tác với CSDL.**

1.Các loại JOIN: có inner join và outer join

- Inner join: lấy những hàng có liên quan nhau giữa hai bảng.

- Left (outer) join: lấy những hàng ở bảng bên trái và những hàng có liên quan giữa hai bảng.

- Right (outer) join: lấy những hàng ở bảng bên phải và những hàng có liên quan giữa hai bảng.

- Full outer join: Lấy tất cả bản ghi ở cả hai bảng. MySQL không hỗ trợ Full outer join thì có thể sử dụng Union All để join hai bảng lại với nhau.

🡪Union: lấy tất cả bản ghi ở cả 2 bảng và loại bỏ những bản ghi trùng nhau.

🡪Union All: lấy tất cả bản ghi ở cả hai bảng và KHÔNG loại bỏ những bản ghi trùng nhau.

2. Phân biệt Group By và Order By:

- Group by: nhóm các hàng với nhau theo các giá trị cột như nhau. Group by thực hiện dựa trên kết quả nhận được từ WHERE>

- Order by: sắp xếp dữ liệu theo thứ tự tăng dần (ASC – mặc định) hoặc giảm dần (DESC) theo một tiêu chí nào đó.

3. Phân biệt WHERE và HAVING:

- Where: điều kiện được sử dụng cho các câu lệnh SQL không chứa các hàm tập hợp (aggregate function). Where đứng trước Group by.

- Having: điều kiện sử dụng chứa các hàm tập hợp (count, sum, avg, min, max). Having đứng sau Group By.

🡪 Trật tự câu lệnh SQL:

SELECT**…(6)**

FROM …(**1**)

WHERE...(2)

GROUP BY…(3)

HAVING …(4)

ORDER BY ….(5).

4.Toán tử điều kiện : AND, OR và NOT.

5. Các toán tử : Between… and; In; Like.

6. Union và Union All: Union lấy tất cả bản ghi ở cả hai bảng, loại bỏ những bản ghi trùng nhau. Còn Union All:không loại bỏ các bản ghi trùng nhau ở cả hai bảng.

7. Exists và In: Exists kiểm tra sự tồn tại của dữ liệu ở trong Database 🡪 Not Exists. Còn In chỉ là 1 toán tử kiểm tra điều kiện sử dụng trong mệnh đề WHERE.

Bài 4: Các hàm thông dụng trong SQL:

1. Count, Sum, Min, Max, AVG.
2. Lcase, Ucase, Now, Concat, Length, Datediff, Substring.

**Bài 5: View and Procedure.**

1.View là một bảng tạm (ảo) truy vấn CSDL được tạo ra từ việc tập hợp các cột của một hoặc nhiều bảng lại với nhau.

Sử dụng view sẽ truy vấn nhanh hơn vì dữ liệu bảng tạm đã được tạo ra sẵn 🡪gọi lại nhiều lần sẽ nhanh hơn.

🡪Tạo view:

**Create view** view\_name

AS

<Select Statement>;

🡪Update: Dữ liệu cập nhật ở View sẽ được cập nhật ở các bảng cơ sở có liên quan.

**Update** view\_name

**Set**  ……………..

**Where** …………;

🡪Không sử dụng Update được với các View được tạo nên từ các câu lệnh Select phức tạp có chứa các từ khóa DISTINCT, GROUP BY, ORDER BY, HAVING, 1 SỐ FUNCTION…

🡪Drop view:

**drop view** view\_name;

2.Stored Procedure: Tập hợp các câu lệnh SQL được sử dụng để thực thi một nhiệm vụ nhất định nào đó (tương tự như method, hàm trong các ngôn ngữ lập trình khác).

**Mục đích sử dụng của Stored Procedure**: có ích cho những tác vụ lặp đi lặp lại, hỗ trợ khai báo biến do người dùng định nghĩa, hoặc thực hiện các tác vụ phức tạp.

**Ưu điểm của Stored Procedure**: tăng tính bảo mật, thực thi tiền biên dịch nên nhanh, giảm thiểu lưu thông trong mô hình client/server, tăng khả năng tái sử dụng.

3.Khai báo biến trong thủ tục:

DECLARE variable\_name data\_type;

DECLARE customer\_id int;

4.Các loại tham số trong Stored Procedure: IN, OUT, INOUT.

🡪IN: tham số truyền giá trị đầu vào cho biến trong stored procedure, đây là chế độ mặc định của Stored Procedure, giá trị của tham số IN trước và sau khi thực hiện Stored Procedure là không thay đổi. ⬄ Tham trị trong các hàm số.

🡪OUT: tham số nhận giá trị trả về của Stored Procedure do người dùng định nghĩa, bằng từ khóa @. Giá trị đầu vào mặc định là NULL.

🡪INOUT: Kết hợp giữa IN và OUT, tức là có thể gán giá trị trước khi truyền vào thủ tục và giá trị của biến sẽ được thay đổi sau khi gọi xong thủ tục.

5.Index: Chỉ mục Index trong SQL là bảng tra cứu đặc biệt mà công cụ tìm kiếm cơ sở dữ liệu có thể sử dụng để tăng nhanh thời gian và hiệu suất truy xuất dữ liệu.

Một chỉ mục Index là một con trỏ chỉ tới từng giá trị xuất hiện trong bảng/cột được đánh chỉ mục. Chỉ mục trong Database có ý nghĩa tương tự như các mục xuất hiện trong mục lục của một cuốn sách.

Index giúp tăng tốc các truy vấn SELECT chứa các mệnh đề WHERE hoặc ORDER, nhưng nó làm chậm việc dữ liệu được thêm vào với các lệnh INSERT/UPDATE vì chỉ mục INDEX cần phải được cập nhật lại.

6.Các loại chỉ mục INDEX:

+(1) *Single-Column Index:* được tạo cho duy nhất 1 cột trong bảng

* **CREATE INDEX** index\_name **ON** table\_name (column\_name);

+(2) *Unique Index:* Là chỉ mục duy nhất, được sử dụng để tăng hiệu suất và đảm bảo tính toàn vẹn của dữ liệu. Một chỉ mục duy nhất không cho phép chèn bất kỳ giá trị trùng lặp nào được chèn vào bảng.

* **CREATE UNIQUE INDEX** index\_name **ON** table\_name (column\_name);

+(3) *Composite Index:* Là chỉ mục kết hợp dành cho hai hoặc nhiều cột trong một bảng.

* CREATE INDEX index\_nam ON table\_name (column\_name\_1, column\_name 2);

+(4) *Implicit Index:* Gọi là Index ngầm định, là chỉ mục được tạo tự động bởi Database Server khi một bảng được tạo ra. Các index ngầm định được tạo tự động cho các ràng buộc **Primary Key** và các ràng buộc **Unique**.

🡪Cú pháp tạo chỉ mục: **CREATE INDEX** index\_name **ON** table\_name (col\_name);

🡪Xóa Index thì sử dụng cú pháp **DROP INDEX** index\_name;

7**.Phân biệt Stored Procedure, Trigger and Function**

🡪Cả stored procedure và function đều là các đối tượng cơ sở dữ liệu chứa một tập các câu lệnh SQL để hoàn thành một tác vụ nào đó.

🡪Một stored procedure có thể sử dụng lại nhiều lần giống như Function.

🡪Stored Procedure có thể trả về giá trị zero, một hoặc nhiều giá trị, trong khi function phải trả về một giá trị duy nhất (có thể là 1 bảng).

🡪Các function chỉ có thể có các tham số đầu vào, trong khi Stored Procedure có tham số đầu vào IN, tham số đầu ra OUT và tham số kết hợp INOUT.

🡪Hàm có thể được gọi từ Stored Procedure, trong khi Stored Procedure không thể được gọi từ hàm.

🡪Stored Procedure không thể được sử dụng trong các câu lệnh Select, trong khi function thì có thể được gọi trong các câu lệnh Select.

🡪Stored Procedure không thể được sử dụng trong các câu lệnh WHERE/HAVING/SELECT trong khi hàm thì có thể được.

**Bài 6: CSS**

1.CSS – Cascading Style Sheets, là ngôn ngữ dùng để tìm và định dạng lại các phần tử được tạo ra bởi ngôn ngữ đánh dấu.

2. Cách nhúng CSS:

🡪Inline style: nhúng trực tiếp trong thẻ HTML bằng thuộc tính style.

🡪Internal style: nhúng trong trang HTML bằng thẻ <style> thường đặt ở phần thẻ <head>

🡪External style: nhúng tệp .css từ bên ngoài, đặt trong thẻ <link> ở phần thẻ <head>

* Độ ưu tiên: inline style 🡪 internal/external style 🡪mặc định của browser.
* Nội dung CSS sẽ được ghi đè bởi các style đặt ở sau. Nếu muốn thuộc tính nào không cho ghi đè thì cần thêm **“!important”** vào thuộc tính đó.

3.Các loại selector:

🡪tag selector

🡪id selector

🡪class selector

* Độ ưu tiên: id selector 🡪class selector 🡪tag selector.

**Bài 7: Thiết kế bố cục trang web**

1.Box model: Mỗi phần tử HTML được xem như là 1 box-model.

Box model gồm có các thành phần: content, padding, border và margin.

Mặc định width của thẻ html chỉ là chiều rộng của phần content của box-model mà thôi.

2.Thuộc tính **position** của box-model:

a.static: vị trí mặc định của thẻ trên trình duyệt. Static không áp dụng các thuộc tính top, bottom, left, right được.

b.relative: vị trí tương đối của thẻ so với vị trí mặc định (static) lúc ban đầu.

c.absolute: vị trí tuyêt đối được so với thẻ cha gần nhất và thẻ cha gần nhất này có thuộc tính position không phải là static.

d.fixed: là vị trí cố định ở trên màn hình viewport khi hiển thị.

3. RWD: Responsive Web Design, là kỹ thuật thiết kế web thích ứng để web có thể hiển thị tốt trên nhiều loại thiết bị với kích cỡ màn hình khác nhau hoặc các loại trình duyệt khác nhau.

4.Grid View: chia màn hình viewport thành 12 cột, mỗi cột chiếm 100/12 = 8.33% chiều rộng màn hình.

5.RWD Media Queries: sử dụng bộ chọn @media only screen and (max-width : 576/768/992/1200px) {} để thiết lập các thuộc tính responsive tương ứng với chiều rộng tối đa của màn hình viewport.

**Bài 8: Boostrap cơ bản.**

1.Phân biệt Framework và Library:

🡪Mỗi framework có nhiều library.

🡪Các framework cung cấp các khung sườn sẵn và các yêu cầu sử dụng nên Dev cần tìm hiểu các khung sườn này để có thể sử dụng đúng chức năng và hiệu quả.

🡪Library: Là công cụ tập hợp các chức năng (function), các class được viết sẵn để có thể tái sử dụng (jQuery, JSTL, JDBC-connector…).

🡪Framework: tập hợp các Library, cung cấp các nguyên tắc cấu trúc ứng dụng => Bắt buộc Dev phải tuân thủ theo khi sử dụng.

🡪So sánh Framework và Library:

+ Giống nhau: cả Framework và Library đều cung cấp các chức năng được viết sẵn để có thể tái sử dụng.

+ Khác nhau: Libs chỉ cung cấp công cụ tiện ích để thoải mái sử dụng, còn Framework tạo ra một bộ khung và cung cấp các hướng dẫn cần tuân thủ theo.

2.Bootstrap: Là 1 frontend framework cung cấp những mẫu thiết kế sẵn dựa trên HTML và CSS. Bootstrap cung cấp khả năng tạo ra các website có tính năng thích ứng (reponsive).

3.Có 2 cách nhúng Bootstrap:

🡪Nhúng qua link CDN (Community Delivered Network): Ưu điểm là folder nhẹ, ít dung lượng nhưng lại phụ thuộc vào bên thứ 3, tốc độ tải trang chậm

🡪Tải bootstrap về và nhúng bằng thẻ <link> trong cặp thẻ <head>.Tốc độ tải trang nhanh, nhưng folder lại nặng.

4.Hệ thống Grid của Bootstrap:

Bootstrap chia màn hình viewport ra thành 12 cột ( mỗi cột chiếm 8.333% chiều rộng khung nhìn.)

Viewport trong Bootstrap được chia thành 4 class:

🡪extra small (xs: col) : màn hình rộng <576px.

🡪 small (col-sm) : màn hình rộng >=576px

🡪medium (col-md): màn hình rộng >=768px.

🡪large (col-lg):màn hình rộng >=992px.

🡪extra large(col-xl): màn hình rộng >=1200px.

Các hàng có class=”row”, các class “container” và “container-fluid”.

🡪Container: màn hình thích ứng có độ rộng 2 bên cố định.

🡪Conainer-fluid: màn hình thích ứng phủ hết độ rộng thiết bị (100%).

**Bài 9: Tổng quan ứng dụng web**

1.Static website và Dynamic website:

🡪Static website: nội dung website không thay đổi theo thời gian, thường được viết bằng các ngôn ngữ phía frontend như HTML, CSS, Boostrap, JavaScript.

🡪Dynamic website: nội dung động, thay đổi theo thời gian, ngữ cảnh, tương tác của người dùng và thường được viết bằng ngôn ngữ backend (Java, C#, Python, PHP…). Dynamic website sẽ được lưu trữ trên một máy chủ web (web server).

2.Giao thức HTTP và HTTPs, SSL.

🡪HTTP: HyperText Transfer Protocol, là giao thức truyền tải siêu văn bản được sử dụng trong world wide web để truyền tải dữ liệu giữa máy chủ web (web server) và máy khách web client (web browser). HTTP có port 80.

🡪HTTPs: HTTP + Secure, thêm tiêu chuẩn bảo mật (mã hóa dữ liệu) theo giấy phép bảo mật SSL (Security Socket Liscence) làm tăng tính bảo mật, an toàn trong lúc truyền tải dữ liệu (thanh toán, thông tin người dùng…). HTTPs có port 433.

🡪Ngoài giao thức HTTP và HTTPs, thì còn có giao thức FTP (File Transfer Protocol) sử dụng để truyền tệp và giao thức SMTP/POP3 (Simple Mail Transfer Protocol/ Post Office Protocol version 3) để truyền tải email điện tử trên internet.

3.Mô hình client/server.

🡪Là mô hình thông dụng trong mạng máy tính, gồm 2 thành phần chính là máy chủ (server) và máy khách (client).

🡪Client gửi request đến Server => Server nhận request, xử lý request và gửi trả lại response.

4.HTTP status code: có 5 loại status code như sau:

🡪1xx: Information, tức là thông tin request đã được chấp nhận và đang được xử lý.

🡪2xx: request được chấp nhận và được xử lý thành công.

🡪3xx: request đang được chuyển hướng và yêu cầu người dùng (client) thực hiện thêm 1 tác vụ nào đó.

🡪4xx: client error, request gửi lên sai cú pháp, không đúng với URLPattern.

🡪5xx: server error, phía server xử lý request thất bại do lỗi server, lỗi lập trình.

5.Java Servlet: Servlet là công nghệ được sử dụng để tạo ra các ứng dụng web động (được triển khai ở phía server). Servlet “mạnh mẽ”, dễ mở rộng, chịu tải tốt và có tính bảo mật (vì được viết trên ngôn ngữ Java).

6.Servlet container: là bộ phận tương tác với Servlet để xử lý các yêu cầu người dùng. Servlet có mục đích chính là:

+ Quản lý vòng đời của các Servlet.

+ Xử lý đa luồng.

+ Object Pool: không cần khởi tạo đối tượng Servlet nhiều lần, tái sử dụng những Servlet trước đó, tương tự như String Pool.

+ Bảo mật.

Vòng đời của Servlet gồm có 5 bước như sau:

(1). Tải class Servlet.

(2). Tạo đối tượng servlet.

(3).Gọi phương thức init().

(4). Gọi phương thức service().

(5). Gọi phương thức destroy().

🡪 Phương thức service() được gọi nhiều lần, còn các phương thức init() và detroy() chỉ được gọi 1 lần duy nhất trong vòng đời mỗi servlet, khi mà Servlet được nạp lên (khi có request đầu tiên từ client gọi đến servlet hiện tại).

🡪Cả 3 phương thức init(), service() và destroy() đều chạy ẩn, chỉ có thể override phương thức service() thông qua 2 phương thức doGet() và doPost() khi tạo servlet mà thôi.

7.Phân biệt request và respone:

🡪Request: yêu cầu từ người dùng (client) gửi đến máy chủ (server) bằng đối tượng request của class HttpServletRequest.

🡪Response: kết quả xử lý request mà máy chủ (server) trả về cho người dùng (client) khi người dùng gửi request trước đó, sử dụng đối tượng response của lớp HttpServletResponse.

8. Phân biệt phương thức doGet() và phương thức doPost().

🡪doGet() xử lý những request được gửi bằng phương thức GET: request là 1 phần của URL, thông tin hiển thị trên trình duyệt nên có thể lưu lại lịch sử và bookmarked được. Số ký tự tối đa là 1024.

🡪doPost() xử lý những request được gửi dưới dạng form bằng phương thức POST (method=”post”): thông tin gửi đi được gói gọn dưới dạng 1 thông điệp, không được hiển thị trên URL và không giới hạn dung lượng gửi đi.

9.Phân biệt forward và redirect:

🡪forward: chuyển hướng có gửi kèm theo dữ liệu ở trang hiện tại đến trang đích

🡪redirect: chuyển hướng đến trang đích nhưng không kèm theo dữ liệu của trang hiện tại.

**Bài 10: JSP và JSTL**

1.Phân biệt JSP và Servlet:

🡪Giống nhau: Đều là ngôn ngữ server side, nằm ở phía backend, có chức năng xử lý dữ liệu động, chịu sự quản lý vòng đời của Servlet container. 🡪 Có 5 giai đoạn trong vòng đời: tải lớp Servlet, tạo đối tượng Servlet, init(), service(), destroy().

🡪Khác nhau: Servlet mạnh về xử lý nghiệp vụ (logic) nên mã HTML nằm trong mã Java. Còn JSP mạnh về xử lý giao diện, nên mã Java nằm trong mã HTML.

🡪Các đối tượng ẩn ở trong JSP:

1. Đối tượng request
2. Đối tượng response
3. Đối tượng out
4. Đối tượng session
5. Đối tượng application
6. Đối tượng config
7. Đối tượng pageContext
8. Đối tượng page
9. Đối tượng exception.

2.JSTL (JSP Standard Tag Library) là thư viện thẻ tiêu chuẩn JSP với mục đích tối ưu lập trình.

JSP có các nhóm thẻ như sau:

1. Nhóm thẻ core (cơ bản): forEach, set, out, redirect, choose/when, if…
2. Nhóm thẻ format (định dạng ngày giờ, ngôn ngữ)
3. Nhóm thẻ SQL (query dữ liệu từ database)
4. Nhóm thẻ function: các hàm hỗ trợ được xây dựng sẵn.
5. XML: làm việc với tài liệu XML.

3.Ý nghĩa của thuộc tính các loại scope trong JSP:

🡪request Scope: biến khai báo chỉ tồn tại trong 1 request. Khi request gửi yêu cầu đi và nhận về response thì biến cũng được giải phóng.

🡪page Scope: biến khai báo scope này tồn tại trong nội bộ page này, không sử dụng được ở bất kỳ page nào khác.

🡪session Scope: biến khai báo scope session tồn tại trong suốt phiên làm việc (từ khi mở đến khi đóng trình duyệt).

🡪application: biến tồn tại trong suốt thời gian từ khi khai báo đến khi tắt server.

**Bài 11: Mô hình MVC**

1.MVC là mô hình Model-View-Controller, chia cấu trúc dự án thành 3 tầng riêng biệt: Model – View – Controller. Trong đó:

+ Model: là phần ứng dụng giao tiếp với dữ liệu (database) và xử lý logic (cùng với Java Bean).

+ View: phần giao diện người dùng, có nhiệm vụ nhận và hiển thị yêu cầu người dùng (jsp, css, HTML, JavaScript.).

+ Controller: kết nối giữa Model và View, nhiệm vụ chính là điều hướng (dựa trên các Servlet).

🡪Lợi ích của kiến trúc MVC:

+ Tái sử dụng.

+ Dễ bảo trì.

+ Dễ mở rộng.

+ Phân chia công việc thành từng phần và có thể bắt đầu thực hiện các công việc đó cùng thời điểm với nhau.

2.Các thành phần của mô hình MVC:  
🡪Model thường được chia nhỏ ra thành JavaBean + BO (Business Object) + DAO (Data Access Object). Trong đó BO kết nối trực tiếp với Controller và DAO làm việc trực tiếp với Database.

🡪Khi request người dùng nhập trên thanh trình duyệt được gửi đi, Controller nhận request và chuyển hướng tới BO, BO gọi DAO để thực hiện. DAO tương tác với Database và JavaBean để lấy dữ liệu và xử lý yêu cầu từ BO. Kết quả xử lý được gửi trả lại cho BO và BO trả lại kết quả đến Controller. Controller gửi kết quả đến View và View hiển thị kết quả lên trình duyệt cho người dùng. Kết thúc một tiến trình request/response.

**Bài 12: JDBC & CRUD**

1.JDBC: Java Database Connectivity, là một API giúp kết nối giữa ứng dụng Java và CSDL (Database). Trình tự làm việc với CSDL bằng JDBC như sau:

(1) Tạo kết nối đến database

Class.forName("**com.mysql.cj.jdbc.Driver**").

Connection **connection = DriverManager.getConnection**(jdbcURL, username, password);

(2) Gửi SQL query đến database sử dụng JDBC driver tương ứng.

Statement **statement = connection.createStatement**(SQL\_query);

(3) JDBC driver kết nối đến database.

Connection **connection = DriverManager.getConnection**(jdbcURL, username, password);

(4) Thực thi câu lệnh query để lấy kết quả trả về (số bản ghi select được, hoặc số bản ghi được insert/update/delete).

ResultSet **resultSet = statement.executeQuery();**

🡪 (*statement.executeUpdate() trả về* ***int***).

(5) Gửi dữ liệu đến ứng dụng thông qua Driver Manager.

ResultSet **resultSet = statement.executeQuery();**

(6) Xử lý dữ liệu trả về.

🡪Sử dụng các phương thức resultSet.getXXX(column\_index **or** column\_name);

1. Đóng (giải phóng) kết nối đến database 🡪close().

**Bài 13: JDBC Query & Transaction.**

1.Phân biệt Statement, PrepareStatement, CallableStatement.

🡪Tất cả đều là Inteface.

🡪Statement: sử dụng để thực hiện các câu lệnh truy vấn tĩnh (không có tham số) và chỉ sử dụng 1 lần: Statement statement = connection.createStatement(SQL\_static\_query);

🡪PreparedStatement: Kế thừa từ Interface Statement, sử dụng để thực hiện các câu lệnh SQL động có tham số, có thể tái sử dụng nhiều lần. PrepareStatement sử dụng có hiệu năng nhanh nhất nên khuyên sử dụng cho cả câu lệnh truy vấn tĩnh.

PrepareStatement statement = connection.prepareStatement(SQL\_dynamic\_query);

statement.setXXX(number\_of\_postion, value\_of\_that\_position);

🡪CallableStatement: Kế thừa từ Interface PreparedStatement, được sử dụng để thực hiện các stored procedure được định nghĩa sẵn trong database.

CallableStatement statement = connection.prepareCall(SQL\_call\_procedure\_query);

2.Transaction: là một tiến trình xử lý các giao dịch, có điểm bắt đầu và điểm kết thúc.

🡪Transaction cho phép thực hiện nhiều câu lệnh thực thi (insert, update, delete…).

🡪Nếu 1 statement trong transaction thất bại thì toàn bộ transaction thất bại.

🡪Cách sử dụng Transaction bằng JDBC:

+ Bắt đầu: connection.setAutoCommit(false);

+ Xử lý statements;

+ Kết thúc: connection.commit();

+ connection.setAutoCommit(true);

+ Nếu một trong những statement trên có lỗi thì trong phần try/catch sẽ thực hiện thao tác: connection.rollback(); // quay trở lại trước khi connection.setAutoCommit(false) ở trên.

3.Gọi thủ tục (stored procedure) bằng JDBC:

+ Kết nối DB với Java Application bằng DriverManager và Driver.

+ Tạo đối tượng connection để kết nối tới Database.

+ Tạo câu lệnh query procedure = “{CALL procedure(tham số);}”

+ Tạo đối tượng để thực hiện câu lệnh truy vấn thủ tục:

CallableStatement statement = connection.prepareCall(SQL\_procedure\_call);

+ Thực hiện truy vấn: ResultSet resultSet = statement.execute ();

+ Đóng (giải phóng) kết nối với Database.

**Bài 14: Dependency Management & Packaging.**

1.Các hình thức đóng gói:

🡪Standalone: Chỉ đóng gói mã nguồn, không bao gồm môi trường thực thi.

🡪Self-container: Bao gồm mã nguồn + JRE (môi trường thực thi Java) để người dùng có thể tự khởi động chương trình.

🡪File đóng gói đã được biên dịch thành file .java và được test sẵn sàng.

2.Phân biệt Ant, Maven và Gradle:

🡪Ant: viết bằng ngôn ngữ XML, chỉ có khả năng đóng gói, biên dịch, kiểm tra mà không có khả năng đóng gói.

🡪Maven: nâng cấp từ Ant, có thêm chức năng quản lý phụ thuộc nhưng vẫn còn hạn chế do viết bởi ngôn ngữ XML nên không thân thiện với Java.

🡪Gradle: có khả năng quản lý phụ thuộc, ngoài biên dịch, kiểm tra và đóng gói, được viết bằng ngôn ngữ Groovy – rất giống Java nên hoạt động tốt. 🡪 Nên Gradle ngày càng phổ biến.

3.Sản phẩm của đóng gói:

🡪JAR: những phần mềm/ứng dụng viết bằng Java.

🡪WAR: những ứng dụng web viết bằng Java.