**CÂU HỎI AUDIT MODULE 3**

**JSP-SERVLET**

# CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên?

CSS – Cascading Style Sheet là 1 ngôn ngữ giúp tìm và định dạng các phần tử được tạo ra bởi HTML. Không được xem là ngôn ngữ lập trình vì nó không cung cấp các câu lệnh if else,v…v…

Có 3 cách nhúng CSS:

- Inline: Sử dụng thuộc tính css trong thẻ HTML

- Internal: Sử dụng thẻ style ở trong thẻ <head>

- External: Sử dụng thẻ <link> để nhúng 1 file css từ bên ngoài

Độ ưu tiên: Inline > (internal ~ external)

Với 2 cách internal và external thì cách nào đc sử dụng gần thẻ hơn thì sẽ ưu tiên cách đó.

# Phân biệt giữa bộ chọn class và bộ chọn id?

|  |  |
| --- | --- |
| Class | Id |
| Sử dụng (.)tên class | Sử dụng (#)tên id |
| Có thể sử dụng nhiều class | Chỉ có thể sử dụng 1 id |

# Trình bày mô hình hộp (box model) trong CSS?

Box model là:

1 chiếc hộp bao quanh phần tử HTML

1 box model bao gồm 4 thành phần:

Content: Nội dung của thẻ

Padding: Là khoảng cách từ content đến border

Border: Viền của thẻ

Margin: Là khoảng cách từ box model này đến box model khác.

Margin sẽ lấy khoảng cách từ border của thẻ này đến border của thẻ kia mà không quan tâm đến margin của phần tử kia.

# Trình bày thuộc tính float, clear?

**Thuộc tính Float** sử dụng để dịch chuyển một phần tử sang góc trái hoặc phải của không gian bao quanh nó, rất cần thiết trong việc định dạng bố cục trang. Theo mặc định, tất cả các phần tử HTML là không **float**.

**Clear** gần như là ngược lại với float. Thuộc tính Clear ngăn chặn thành phần A chiếm vùng không gian của thành phần B (với thành phần B là thành phần sử dụng float). Đôi khi không muốn float ở một số tình huống nào đó ta sẽ dùng clear để khắc chế.

# Thuộc tính position dùng để làm gì?

Position trong Css được sử dụng để định vị vị trí hiển thị của các phần tử HTML

Phân biệt các giá trị: static, absolute, relative và fixed?

|  |  |  |  |
| --- | --- | --- | --- |
| **Static** | **Absolute** | **Relative** | **Fixed** |
| Vị trí mặc định khi thẻ đc tạo ra, ko bị ảnh hưởng bởi top, left, bottom, right | So tuyệt đối với thẻ cha | Gần giống với static nhưng bị ảnh hưởng bởi top, left, bottom, right | Cố định so với màn hình, ko bị ảnh hưởng bởi scroll |

# Responsive Web Design là gì?

Là để thể hiện trang web trên tất cả các thiết bị với đầy đủ nội dung và ko xuất hiện thanh scrollbar nằm ngang.

Grid-View: Áp dụng để thiết kế 1 trang web theo dạng lưới -> có nhiều dòng và nhiều cột.

# Làm thế nào để tạo 1 trang web responsive?

Sử dụng @Media hoặc grid-view + responsive.

# Bootstrap là gì?

Framework HTML, CSS và JS cho phép ng dùng dễ dàng thiết kế website theo 1 chuẩn nhất định.

Framework là

* Có 2 cách nhúng:

- Download => nhúng vào HTML bằng external

- Sử dụng CDN để nhúng (external)

# Một số thành phần thông dụng ở Bootstrap?

|  |  |
| --- | --- |
| Badge | Thường dùng cho hiển thị số lượng thông báo, tin nhắn |
| Carrousel | Dùng chạy quảng cáo (Web đặt vé film: Top film có doanh thu cao 1’) |
| Card | Thường dùng cho list (Sản phẩm, giáo viên, học sinh) |
| Modal | Hiển thị hộp thoại thông báo xác nhận |
| Navbar | Thanh điều hướng (Thường nằm ở header), ko cần để trong container |
| Spinner | Loading |
| Hỗ trợ gridview | Thiết kế gridview phải đc đặt trong container hoặc container-fluid |

# Phân biệt Static web và Dynamic web?

|  |  |
| --- | --- |
| **Static web** | **Dynamic web** |
| Xây dựng dựa trên HTML, CSS và JS | Xây dựng dựa trên 2 dòng ngôn ngữ:   * Back end: Java (Python, PHP, C#) * Front end: HTML, CSS, JS (Angular) |
| Ko thay đổi | Data luôn luôn thay đổi và cập nhật liên tục |

# Phân biệt 2 phương thức GET/POST

|  |  |
| --- | --- |
| **Get** | **Post** |
| Data hiển thị ở URL | Ko hiển thị ở URL |
| Kém bảo mật | Bảo mật ( được mã hóa) |
| Nhanh hơn | Chậm hơn |
| Dữ liệu gởi đi là character | Có thể gởi hình ảnh, video |
| Dung lượng giới hạn là 2048 kí tự ( tùy vào trình duyệt) | Ko giới hạn ( tùy vào trình duyệt) |

# Phân biệt request/response

|  |  |
| --- | --- |
| **Request** | **Response** |
| Yêu cầu từ client (Người dùng) => server | Phản hồi của server => client |

# Phân biệt forward/redirect

|  |  |
| --- | --- |
| **Forward** | **Redirect** |
| Chuyển trang nhưng kèm theo dữ liệu | Chuyển trang nhưng ko kèm theo dữ liệu |

# JSP là gì?

JSP( Java Server Page) là công nghệ được sử dụng để tạo web động, được viết dựa trên HTML và **có thể nhúng Java để đổ dữ liệu.**

# Nói: “Về bản chất JSP cũng chính là Servlet đúng hay sai? Giải thích?

Nguồn gốc câu nói JSP là Servlet: JSP sẽ phải biên dịch ra Servlet rồi mới chạy được.

So sánh giống và khác nhau giữa JSP và Servlet ?

* Giống: Cả 2 đều dùng đc cả HTML và Java.
* Khác:

|  |  |
| --- | --- |
| **JSP** | **Servlet** |
| Ưu tiên trong phần xử lý giao diện | Ưu tiên trong phần xử lý nghiệp vụ, kiểm tra tính hợp lệ (validate) |

# JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ?

JSTL (JSP standard tag library) là bộ thư viện cung cấp các thẻ đc xây dựng dựa trên ngôn ngữ java.

Một số thẻ hay sử dụng:

1. **Vòng lặp:**  
   <c:foreach var=”” items=””>

Trong đó: Var: Đại diện cho 1 phần tử của list với mỗi vòng lặp.

Items: chứa list gởi từ servlet sang.

1. **In giá trị**:

<c:out>: in giá trị.

<${}>: in giá trị nhưng sẽ dịch ra HTML.

1. **Điều kiện:**

<c:if test=””>

Test là biểu thức trả về true/false

* <c:choose> ~ if else bậc thang.

+ <c:when> ~ if: điều kiện.

+ <c:otherwise> ~ else.

# Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC?

MVC có 3 thành phần chính:

* Model: (Java) Xử lý nghiệp vụ (service), kết nối DB (repository), biểu diễn dữ liệu (model, entity)
* View: Là thành phần giao diện để ng dùng tương tác với hệ thống (JSP)
* Controller: Thành phần điều hướng, nhận và gửi dữ liệu từ view xuống model (Servlet)

# Luồng di chuyển trong mô hình MVC

Khi có 1 request từ người dùng :

* Controller sẽ nhận request và gửi dữ liệu sang model giải quyết.
* Model sẽ thực hiện validate , tính toán dữ liệu ( nếu cần truy cập Database thì Model sẽ kết nối đến DB để xử lí).
* Sau đó gửi dữ liệu về lại Controller.
* Controller sẽ Render Data đến View.

View sẽ hiển thị thông tin đến người dùng.

# Sự khác nhau giữa MVC1 và MVC2

MVC 1 thì View và Controller đều được viết bằng JSP => khó phát triển và mở rộng

MVC 2 thì Controller đc viết bằng Servlet

# Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC?

* B1: Tạo connection
* B2: Tạo đối tượng Statement / preparedStatement / callableStatement.
* B3: Tạo đối tượng ResultSet, executeQuery và chạy vòng lặp while để lấy giá trị (Đối với trường hợp Select). Sử dụng Statement / preparedStatement / callableStatement.set + Statement / preparedStatement / callableStatement.executeUpdate đối với trường hợp Update, Insert, Delete.

# Phân biệt các loại Statement ?

Statement & preparedStatement dùng để tạo câu lệnh truy vấn ở Java.

CallableStatement dùng để gọi 1 stored procedure đc tạo từ DB.

|  |  |
| --- | --- |
| Statement | PreparedStatement |
| Lỗ hổng bảo mật (SQL injection) | Tham số là dấu ? nên sẽ ngăn đc SQL injection |
| Dùng để tạo câu lệnh truy vấn tĩnh | Dùng để tạo câu lệnh truy vấn động (Truyền tham số ?) |
|  | PreparedStatement là lớp kế thừa từ lớp cha Statement |

# Kỹ thuật sử dụng transaction bằng JDBC là như thế nào?

Auto Commit set giá trị là False sau đó nếu tất cả các bước đều đúng, thì sẽ Commit, nếu 1 trong các bước bị sai, sẽ tự động roll back.

# Các tính chất của Transaction?

- Atomicity: Tất cả hoặc ko có gì.

- Consistency: Tính nhất quán giữa Servlet và Database.

- Isolation: Nếu có 2 transaction cùng cập nhật 1 dữ liệu thì MYSQL sẽ đảm bảo thực hiện tuần tự mà ko bị dẫm chân lên nhau (đồng bộ).

- Durability: Dữ liệu khi đã commit thì trở nên cố định và luôn như vậy.

# Phương pháp gọi 1 SP ở JDBC?

Sử dụng callableStatement.prepareCall (query)

|  |  |
| --- | --- |
| **Query** | |
| **Cú pháp** | **Stores Procedures** |
| { call procedure\_name() } | Không có đối số và không có giá trị trả về |
| { call procedure\_name(?,?) } | Chấp nhận 2 đối số và không có giá trị trả về |
| {?= call procedure\_name() } | Không có đối số và có giá trị trả về |
| {?= call procedure\_name(?) } | Chấp nhận 1 đối số và có giá trị trả về |

# Phân biệt Ant, Maven, Gradle?

* Ant: Muốn sử dụng 1 thư viện nào đó thì phải import thủ công (down về tự import vào project)
* Maven: Ra đời nhằm khắc phục các nhược điểm của Ant. Sử dụng file pom.xml để quản lý thư viện, tuy nhiên file này còn dài dòng do sử dụng xml bao gồm rất nhiều thẻ.
* Gradle: Khắc phục đc nhược điểm của Ant và Maven, cũng như hỗ trợ nhiều ngôn ngữ. Gradle sử dụng file build.gradle để cấu hình (sử dụng groovy nên ngắn gọn hơn)