|  |  |
| --- | --- |
| 1.Cơ sở dữ liệu là gì? | Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng chạy cùng một lúc với những mục đích khác nhau. |
| 2. Phân biệt các câu lệnh DML & DDL? | **DDL giúp thay đổi cấu trúc của cơ sở dữ liệu trong khi DML giúp quản lý dữ liệu trong cơ sở dữ liệu.**  DDL: create, alter, drop. “bảng”  DML: insert, update, delete, select. “dữ liệu của bảng” |
| 3.Trình bày về mô hình thực thể mối quan hệ (ERD - Entity-Relationship Diagram)? Trong mô hình thực thể mối quan hệ có những thành phần nào? | ERD là một sơ đồ, thể hiện các thực thể có trong database, và mối quan hệ giữa chúng với nhau.  Mục tiêu của mô hình ERD trong quá trình thiết kế cơ sở dữ liệu đó là phân tích dữ liệu, xác định các đơn vị thông tin cơ bản cần thiết của tổ chức, mô tả cấu trúc và mối liên hệ giữa chúng.  Các thành phần có trong ERD:   * Thực thể và tập thực thể * Thuộc tính * Mối quan hệ và tập mối quan hệ |
| 4. Các bước chuyển đổi từ ERD -> Table? | B1: Biến đổi thực thể thành table theo nguyên tắc:  -Thuộc tính khóa của thực thể là khóa chính của table  - Các thuộc tính thông thường của thực thể là thuộc tính của table  B2: Biến đổi thuộc tính đa trị   * Tách thuộc tính đa trị thành bảng mới, xác định mối quan hệ giữa bảng mới và bảng hiện tại   B3: Biến đổi quan hệ 1:1   * Đặt thuộc tính khóa ở phía tùy chọn sang bắt buộc   B4: Biến đổi mối quan hệ 1-n:   * Đặt khóa chính bên 1 là khóa ngoại bên n   B5: Biến đổi mối quan hệ n-n:   * Tách thành bảng mới, có khóa chính là tập hợp 1 khóa của 2 thực thể tham gia vào mối quan hệ => gọi là khóa phức hợp |
| 5. Các bước chuẩn hóa dữ liệu? | **+ Dạng chuẩn 1NF:**  - Các thuộc tính của bảng phải là nguyên tố  - Giá trị của các thuộc tính trên các hàng phải là đơn trị, không chứa nhóm lặp  - Không có một thuộc tính nào có giá trị có thể tính toán được từ một thuộc tính khác  **+ Dạng chuẩn 2NF:**  **Quy tắc** chuẩn hóa từ chuẩn 1NF thành 2NF:  **Bước 1:** Loại bỏ các thuộc tính không khóa phụ thuộc vào một bộ phận khóa chính và tách ra thành một bảng riêng, khóa chính của bảng là bộ phận của khóa mà chúng phụ thuộc vào.  **Bước 2:** Các thuộc tính còn lại lập thành một quan hệ, khóa chính của nó là khóa chính ban đầu.  **+ Dạng chuẩn 3NF:**  ***Điều kiện:***  Phải đạt chuẩn 2NF  Mọi thuộc tính không khóa phụ thuộc bắc cầu vào thuộc tính khóa (nghĩa là tất cả các thuộc tính không khóa phải được suy ra trực tiếp từ thuộc tính khóa)  Quy tắc chuẩn hóa từ 2NF thành 3NF:  **Bước 1:** Loại bỏ các thuộc tính phụ thuộc bắc cầu ra khỏi quan hệ và tách chúng thành quan hệ riêng có khóa chính là thuộc tính bắc cầu.  **Bước 2:** Các thuộc tính còn lại lập thành một quan hệ có khóa chính là khóa ban đầu. |
| 6. Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL? | Constraint là những quy tắc được áp dụng trên các cột dữ liệu, trên bảng. Được sử dụng để kiểm tra tính hợp lệ của dữ liệu vào, đảm bảo tính chính xác, tính toàn vẹn của dữ liệu.  Các loại rang buộc: |
| 7.Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì? | * Khóa chính (Primary Key, hay ràng buộc khóa chính) được sử dụng để định danh duy nhất mỗi bản ghi (dòng) trong bảng của cơ sở dữ liệu.   Khóa chính có thể tạo ra dựa trên một cột hoặc nhiều cột (được gọi là khoá chính tổ hợp) của table. |
| 8. Khóa ngoại (foreign key) là gì? | Khoá ngoại (foreign key - viết ngắn gọn là FK) là cơ chế để tạo liên kết giữa các bảng trong cùng một cơ sở dữ liệu. |
| 9.Phân biệt DELETE và TRUNCATE? | Cú pháp: Delete: DELETE FROM table\_name;  Truncate: TRUNCATE TABLE table\_name;  Có một điểm khác nhau nhỏ giữa lệnh DELETE và lệnh TRUNCATE. Lệnh **DELETE** chỉ xóa các hàng từ bảng dựa trên điều kiện đã được định nghĩa bởi mệnh đề WHERE hoặc xóa tất cả các hàng từ bảng nếu điều kiện này không được xác định. Nhưng nó **không** giải phóng bộ nhớ chứa bảng đó.  Trong khi đó, lệnh TRUNCATE xóa tất cả các hàng từ bảng và **giải phóng bộ nhớ** chứa bảng đó. Để xóa tất cả các hàng từ bảng SINHVIEN, truy vấn như sau:   * TRUNCATE TABLE SINHVIEN; * Dữ liệu bị xóa bởi DELETE có thể restored được còn TRUNCATE thì không. * Tùy vào từng trường hợp thì bạn sẽ dùng DELETE hoặc TRUNCATE, TRUNCATE sẽ luôn nhanh hơn vì nó tiêu tốn ít bộ nhớ hơn, các bạn có thể cân nhắc điều này khi cần xóa 1 bảng lớn vs nhiều record. |
| 10.Các loại JOIN? Phân biệt? | + Inner join: Lấy phần tử chung giữa 2 bảng  + Outer join:   * Left join: Lấy phần chung và toàn bộ bảng bên trái * Right join: Lấy phần chung và toàn bộ bảng bên phải * Full join lấy hết |
| 11. Phân biệt JOIN và UNION? |  |
| 12. Phân biệt UNION và UNION ALL? | UNION có tác dụng kết hợp hai lệnh SELECT và loại bỏ đi những dòng bị trùng lặp chỉ để lại một dòng duy nhất, tuy nhiên không phải lệnh nào cũng gộp được mà phải thoải điều kiện sau:   * Tên của các column phải giống nhau * Thứ tự các column phải giống nhau * Tổng các column phải bằng nhau   UNION ALL có cú pháp và tác dụng tương tự như UNION, chỉ có điều là lệnh UNION ALL sẽ giữ lại tất cả dữ liệu kể cả những dòng bị trùng lặp. |
| 13. Phân biệt EXIST và IN? | Còn về IN và EXIST thì cách sử dụng không hề khác nhau. Điểm khác biệt là về performance.  Exist nhanh hơn  In sẽ tìm ra tất cả các bộ thỏa yêu cầu câu truy vấn con rồi mới so sánh với câu truy vấn mẹ Exists sẽ quét từng bộ ở câu truy vấn con và so sánh trực tiếp với yêu cầu của mẹ, nếu không thỏa là loại trừ ngay |
| 14.  Phân biệt AND và OR? | AND cho phép sử dụng nhiều điều kiện trong mệnh đề WHERE của câu lệnh SQL. Như ở bài trước bạn đã thấy, chúng ta chỉ có thể sử dụng một điều kiện trong WHERE, khi kết hợp với [SELECT](https://quantrimang.com/lenh-select-trong-sql-143756). Nhưng nếu có AND bạn có thể thêm vào các điều kiện khác để đảm bảo lấy được dữ liệu chuẩn xác hơn. Dữ liệu lấy ra phải thỏa mãn cùng lúc tất cả các điều kiện có trong mệnh đề WHERE.  Toán tử OR được sử dụng để kết hợp nhiều điều kiện trong mệnh đề WHERE của lệnh SQL. Khi sử dụng OR trong WHERE kết hợp với SELECT thì dữ liệu lấy ra chỉ cần thỏa mãn một trong các điều kiện được liệt kê. |
| 15.  Phân biệt GROUP BY và ORDER BY? | Vậy nhận thấy mệnh đề group by là một phần không bắt buộc của câu lệnh select, được sử dụng để nhóm các tập các hàng có cùng một giá trị vào thành một nhóm, mỗi nhóm đó chỉ trả về một hàng.  Các hàm tập hợp (SUM, MAX, MIN, COUNT, AVG) cho phép tính toán một tập các hàng và trả về một giá trị duy nhất. Mệnh đề group by thường được sử dụng với một hàm tập để thực hiện việc tính toán và trả về một giá trị cho mỗi nhóm.  Mệnh đề ORDER BY được dùng để lọc bản ghi trong bộ kết quả trả về của [SQL Server](https://quantrimang.com/sql-server).  **ASC** - tùy chọn. Lọc kết quả theo thứ tự tăng dần của biểu thức (mặc định nếu không chỉ định rõ).  **DESC** - tùy chọn. Lọc kết quả theo thứ tự giảm dần của biểu thức.  **Lưu ý:**  Nếu không chọn cụ thể ASC hay DESC trong mệnh đề ORDER BY thì kết quả sẽ mặc định được sắp xếp theo thứ tự tăng dần, tương đương với ORDER BY “biểu thức” ASC. |
| 16. Muốn nối chuỗi trong MySQL thì sử dụng gì? | để nối chuỗi trong MYSQL thì ta phải làm thế nào? Đơn giản là ta sẽ dùng hàm **Concat trong MYSQL**  Hàm concat có tác dụng là nối các chuỗi lại với nhau với cú pháp như sau:   |  |  | | --- | --- | | 1 | concat(str1, str2, str2, ... , strn) | |
| 17. Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào? | SELECT  AVG(ten\_cot\_diem) ten\_bang |
| 18 | Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào? |
|  | SELECT name\_student, avg(mark\_column) FROM student GROUP BY name\_student; |
| 19 | Yêu cầu hiển thị tên sinh viên và điểm trung bình tương ứng, chỉ hiển thị các bạn có điểm trung bình lớn hơn 5 thì làm thế nào? |
|  | SELECT name\_student, avg(mark\_column) dtb FROM student  HAVING dtb > 5  GROUP BY name\_student; |
| 20 | Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID)? |
|  | SELECT st.name\_student , cls.class\_name FROM student st  JOIN class cls ON cls.class\_id = st.class\_id; |
| 21 | Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi? |
|  | SELECT \* FROM student ORDER BY mark; |
| 22 | Hiển thị danh sách sinh viên theo thứ tự giảm dần của họ tên, nếu họ tên trùng nhau thì sắp xếp theo tăng dần của tuổi? |
|  | SELECT \* FROM student ORDER BY name\_student DESC, age; |
| 23 | Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’? |
|  | SELECT \* FROM student WHERE student\_name LIKE ‘Le%’ AND student\_name LIKE ‘%Hai’; |
| 24 | Hiển thị các sinh viên có tên bắt đầu là ‘C’? |
|  | SELECT \* FROM student  HAVING (REVERSE(SUBSTRING(REVERSE(student\_name), 1, INSTR(REVERSE(student\_name,' ')) LIKE ‘C%’; |
| 25 | Làm sao để có thể tăng tốc độ truy vấn? Nhược điểm là gì? |
|  | - **Chỉ mục (Index):** là bảng tra cứu đặc biệt mà Database Search Engine có thể sử dụng để tăng thời gian và hiệu suất truy vấn dữ liệu.  - Nhược điểm :  Index làm tăng hiệu năng của lệnh **SELECT** nhưng lại làm giảm hiệu năng của lệnh **INSERT**, **UPDATE** và **DELETE** bởi vì khi database thay đổi thì chúng ta cần thực hiện lại việc đánh index (Ví A có index là 0, B có index là 1, khi xóa A đi thì B có index là 0, các trường phía sau cũng thay đổi). Chỉ nên index những trường có kiểu dữ liệu số. Những kiểu dữ liệu khác nếu không phải là đặc biệt, hoặc ko phải tìm kiếm nhiều thì ko nên index. |
| 26 | Các dạng tham số ở SP? |
|  | * **IN**: Đây là chế độ mặc định (*nghĩa là nếu bạn không định nghĩa loại nào thì nó sẽ hiểu là IN*). Khi bạn sử dụng mức này thì nó sẽ được bảo vệ an toàn, có nghĩa là sẽ không bị thay đổi nếu như trong Procedure có tác động đến * **OUT**: Chế độ này nếu như trong Procedure có tác động thay đổi thì nó sẽ thay đổi theo. Nhưng có điều đặc biệt là dù trước khi truyền vào mà bạn gán giá trị cho biến đó thì vẫn sẽ không nhận được vì mặc định nó luôn hiểu giá trị truyền vào là NULL. * **INOUT**: Đây là sự kết hợp giữa **IN** và **OUT**. Nghĩa là **có thể gán giá trị trước** và **có thể bị thay đổi** nếu trong Procedure có tác động tới |
| 27 | Phân biệt SP, Trigger, Function? |
|  | **Procedure** | | **Funtion** | **Trigger** |
| Tham số | IN, OUT, INOUT | | IN | Không có |
| Cách gọi | CALL sp\_name | | SELECT, UPDATE function\_name | Tự động được gọi (khi bảng theo dõi xảy ra 1 sự kiện) |
| Giá trị trả về | Có thể có một hoặc nhiều kết quả SELECT và các tham số OUT | | Trả về một giá trị duy nhất thông qua RETURN |  |
| Gọi thủ tục, hàm | Có thể gọi SP và hàm khác | | Chỉ có thể gọi các hàm khác |  |
| 28 | CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên? |
|  | - CSS là chữ viết tắt của **Cascading Style Sheets**, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web  - Độ ưu tiên trong CSS:  + **!important**: cao nhất  + **Internal & External**: có độ ưu tiên ngang nhau, nên sẽ xét độ ưu tiên theo cách nó được gọi (ai gọi sau thì được ưu tiên hơn)  + **Inline** (1000)  + **#id** (100)  + .**class** (10)  + **tag** (1) |
| 29 | Phân biệt giữa bộ chọn class và bộ chọn id? |
|  | |  |  | | --- | --- | | **ID** | **Class** | | - Khai báo bằng “#” | - Khai báo bằng “.” | | - Là duy nhất, không được trùng nhau, chỉ sử dụng cho một phần tử duy nhất | - Có thể trùng nhau, có thể sử dụng cho nhiều class | |
| 30 | Trình bày mô hình hộp (box model) trong CSS? |
|  | - **Box model** là hộp bao quanh 1 thẻ html  - Có 4 thành phần:  + **content**: nội dung  + **padding**: khoảng cách từ content => border  + **border**: độ dày đường viền  + **margin**: khoảng cách từ border của thẻ đến border của thẻ khác |
| 31 | Trình bày thuộc tính float, clear? |
|  | - Float:  **Thuộc tính Float** sử dụng để 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*.  Thuộc tính *Float* có thể có một trong các giá trị sau:   * **left**: Cố định phần tử về bên trái. * **right:** Cố định phần tử về bên phải. * **none:** Nằm tại chính vị trí của nó (trạng thái bình thường). * **inherit** (kế thừa): Phần tử kế thừa giá trị từ float cha.   - Clear:  **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.  Nói một cách đơn giản, *clear* sử dụng để giải quyết vấn đề trong mục **chú ý** của thuộc tính *float* phía trên.  Thuộc tính *float* có thể có một trong các giá trị sau:   * **left:** Tràn về phía bên trái. * **right:** Tràn về phía bên phải. * **none:** Cho phép tràn lên cả hai phía (mặc định). * **both:** Không cho phép tràn về bên nào * **inherit** (kế thừa): Phần tử kế thừa giá trị từ float cha.   Cách phổ biến nhất để sử dụng thuộc tính *Clear* là dùng sau khi bạn đã sử dụng thuộc tính *Float* trên một phần tử. Nếu một phần tử được float sang bên trái, thì bạn nên clear bên trái. Phần tử *Float* của bạn sẽ tiếp tục nổi, nhưng phần tử bị xóa sẽ xuất hiện bên dưới nó trên trang web. |
| 32 | Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed? |
|  | - **Position** là thuộc tính của CSS để xác định cách thức hiện thị vị trí của thẻ HTML  **+ Static:** Đây được xem là giá trị hiển thị Position trong css một cách mặc định (default), các thành phần sẽ nằm theo thứ tự của văn bản**.**Không bị ảnh hưởng bởi thuộc tính top, left, right, bottom  **+ Relative:**Định vị trí tương đối cho các thành phần, không gây ảnh hưởng tới vị trí ban đầu hay các thành phần khác. Bị ảnh hưởng bởi thuộc tính top, left, right, bottom  **+ Absolute:**Giá trị này sẽ giúp định vị trí tuyệt đối cho thành phần theo thành phần bao ngoài, hoặc ít nhất là theo cửa sổ trình duyệt. **Fixed:**Định vị trí tương đối cho thành phần theo cửa sổ trình duyệt.  **+ Fixed:**Định vị và giúp cho phần tử luôn cố định một chỗ, ví dụ như khi bạn scroll trình duyệt chẳng hạn, phần tử sẽ không thay đổi.  **+ Inherit:**Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài). |
| 33 | Responsive Web Design là gì? |
|  | **Responsive web design (RWD) hay còn gọi: Thiết kế đáp ứng**là kiểu mẫu phong cách thiết kế với giao diện. Bố cục website thể hiện chuyên nghiệp và mang tính thẩm mỹ với độ hiển thị nội dung có giãn phù hợp trên tất cả màn hình của các thiết bị như: desktop, laptop, tablet, mobile,… Duy trì sự hiển thị nội dung nhất quán thẩm mỹ trên mọi chế độ phân giải**.** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 34 | | | Làm thế nào để tạo 1 trang web responsive? | |
|  | | | 1. Khai báo meta viewport  * meta viewport nghĩa là một thẻ thiết lập cho trình duyệt hiển thị tương ứng với kích thước màn hình. Chẳng hạn như ví dụ trên, có nghĩa là bạn sẽ thiết lập trình duyệt hiển thị cố định và tương ứng trên tất cả các thiết bị dựa vào chiều rộng của thiết bị (device-width) và không cho phép người dùng phóng to (thiết lập initial-scale với giá trị cố định là 1). Đây là thẻ mình khuyến khích bạn sử dụng cho toàn bộ các dự án Responsive.  1. Sử dụng grid view (để chia dạng lưới 12 cột) + media querise:  * Media Queries được xây dựng như một cách mở rộng các đối tượng. Nó giúp chúng ta đặt các kiểu style khác nhau cho đối tượng trên các trình duyệt và tỷ lệ khung hình khác nhau. | |
| 35 | | | Bootstrap là gì? | |
|  | | | Bootstrap là một front-end framework miễn phí để phát triển web nhanh hơn và dễ dàng hơn.  Bootstrap bao gồm các mẫu thiết kế dựa trên html và css.  Bootstrap cung cấp khả năng tạo các thiết kế responsive. | |
| 36 | | | Một số thành phần thông dụng ở Bootstrap? | |
|  | | | * Navbar:thường dùng để làm header * Dropdown menu: thường dùng để làm menu tùy chọn * Carousel: dùng để hiển thị dạng slide show * Card: thường dùng cho chức năng danh sách. * Container/container-fluid: dùng để thiết kế phần thân body chứa thông tin * … | |
| 37 | | | Phân biệt Static web và Dynamic web? | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | WEB TĨNH | | | WEB ĐỘNG | | Về chức năng | Không thể thay đổi hay chỉnh sửa phần nội dung khi đã đăng lên website. | Có thể quản lý và chỉnh sửa nội dung bên trong website | | Về ngôn ngữ lập trình | Sử dụng duy nhất ngôn ngữ HTML. | Đa dạng ngôn ngữ như ASP.NET, PHP kết hợp với cơ sở dữ liệu SQL server, MySQL... | | Về khả năng tương tác với người dùng | Người dùng không thể thực hiện các thao tác trên website như trò chuyện hay gửi bất kỳ phản hồi nào lên nó. | Khách hàng có thể thoải mái trao đổi thông tin với chủ sở hữu website và những vị khách khác bằng những thao tác cực kỳ đơn giản. | | Về việc ứng dụng | Thường được dùng cho những trang web nhỏ, có phần nội dung nhất định, ít bị thay đổi. | Được ứng dụng trong các website phục vụ cho lĩnh vực tin tức, blog, các website giới thiệu doanh nghiệp hay các website sàn thương mại điện tử... Nói chung, [web động](https://thietkewebso.com/tim-kiem/?q=web+%C4%91%E1%BB%99ng) có khả năng ứng dụng cao hơn web tĩnh. | | Về chi phí bảo trì - nâng cấp | Chỉ cần bỏ vốn đầu tư một lần mà sử dụng được lâu dài.  Web tĩnh có giao diện thân thiện với các công cụ tìm kiếm hơn web động.  Tuy nhiên, việc bảo trì - nâng cấp và quản lý nội dung đối với [web tĩnh](https://thietkewebso.com/tim-kiem/?q=web+t%C4%A9nh) tương đối phức tạp, khó thực hiện. | Tuy chi phí đầu tư ban đầu cho [web động](https://thietkewebso.com/tim-kiem/?q=web+%C4%91%E1%BB%99ng) khá lớn, nhưng quá trình nâng cấp - bảo trì lại vô cùng đơn giản và dễ thực hiện.  Thích hợp đối với các doanh nghiệp lớn | | | | | | |
| 38 | | | Phân biệt 2 phương thức GET/POST | |
| |  |  | | --- | --- | | GET | POST | | Phương thức GET gửi thông tin người dùng đã được mã hóa được phụ thêm vào yêu cầu trang, truyền thông tin thông qua url. | Phương thức POST truyền thông tin thông qua HTTP header | | Dữ liệu của METHOD GET gửi đi thì hiện trên thanh địa chỉ (URL) của trình duyệt. | Dữ liệu được gửi đi với METHOD POST thì không hiển thị trên thanh URL | | HTTP GET có thể được cache bởi trình duyệt | HTTP POST không cache bởi trình duyệt | | HTTP GET có thể duy trì bởi lịch sử đó cũng là lý do mà người dùng có thê bookmark được. | HTTP POST không thể duy trì bởi lịch sử đó cũng là lý do mà người dùng không thê bookmark HTTP POST được. | | Không bảo mật | Bảo mật | | Thực thi nhanh hơn POST vì những dữ liệu gửi đi luôn được webbrowser cached lại. | Thực thi chậm hơn GET | | phương thức GET ứng với cùng một yêu cầu đó webbrowser sẽ xem trong cached có kết quả tương ứng với yêu cầu đó không và trả về ngay không cần phải thực thi các yêu cầu đó ở phía server. | Khi dùng phương thức POST thì server luôn thực thi và trả về kết quả cho client | | Phương thức GET được giới hạn gửi tối đa chỉ 2048 ký tự | Phương thức POST không có bất kì hạn chế nào về kích thước dữ liệu sẽ gửi. | | Không gửi được nhị phân. | Phương thức POST có thể sử dụng để gửi ASCII cũng như dữ liệu nhị phân | | Không bao giờ sử dụng phương thức GET nếu gửi password hoặc thông tin nhay cảm lên Server. | Dữ liệu gửi bởi phương thức POST thông qua HTTP header, vì vậy việc bảo mật phụ thuộc vào giao thức HTTP. Bằng việc sử dụng Secure HTTP, bạn có thể chắc chắn rằng thông tin của mình là an toàn. | | Dữ liệu gửi bởi phương thức GET có thể được truy cập bằng cách sử dụng biến môi trường QUERYSTRING. | Không thể | | Gửi lại form Với form gửi đi bằng phương thức GET bạn có thể gửi lại bằng cách bấm phím F5 hoặc Ctrl + R | nếu bạn muốn thực hiện việc gửi lại dữ liệu của form thì trình duyệt sẽ hiển thị một hộp thoại cảnh báo. Trở lại trang trước | | Dữ liệu gửi đi được lưu lại trong lịch sử web và có thể xem lại | với POST thì bạn sẽ thấy một trang trống. | | đối với dữ liệu ít thay đổi thường dùng phương thức GET để truy xuất và xử lý nhanh hơn. | Đối với những dữ liệu luôn được thay đổi thì thường sử dụng phương thức POST | | dữ liệu không cần bảo mật thì dùng phương thức GET | dữ liệu bảo mật thì dùng phương thức POST. | | | | | | |
| 39 | | | Phân biệt request/response | |
| |  |  | | --- | --- | | REQUEST | RESPONE | | Request có thể hiểu nhanh là thông tin gửi từ client lên server, lúc này phía server sẽ phân tích yêu cầu và sẽ gửi luồng xử lý tới vị trí vị trí lưu trữ của mã nguồn PHP (hoặc mã nguồn bất kì) và nhiệm vụ của các mã nguồn là tiếp nhận yêu cầu, phân tích request đó và trả kết quả lại cho client. | Reponse là dữ liệu mà server trả về cho client. Ví dụ khi bạn nhập vào địa chỉ domain thì kết quả trả về (*response*) chính là giao diện của website và các thông tin của header. Như vậy dữ liệu mà server trả về là những đoạn mã HTML kèm theo các thông tin của header | | | | | | |
| 40 | | | Phân biệt forward/redirect | |
| |  |  | | --- | --- | | **FORWARD** | **REDIRECT** | | Forward là chuyển tiếp từ webserver sang trang khác kèm theo dữ liệu | Redirect là chuyển hướng sang trang khác(không kèm theo dữ liệu) | | Khi một yêu cầu (request) của trình duyệt gửi tới một **Servlet**, nó có thể chuyển tiếp yêu cầu tới một trang khác (hoặc một servlet khác). Địa chỉ trên trình duyệt của người dùng vẫn là đường dẫn của trang đầu tiên, nhưng nội dung của trang do trang được chuyển tiếp tới tạo ra. | Khi một yêu cầu (request) từ phía người dùng tới một Servlet (Trang A), servlet này có thể chuyển yêu cầu này tới một trang khác (Trang B), và kết thúc nhiệm vụ của nó. Trang được chuyển hướng tới có thể là trang trong ứng dụng của bạn, hoặc có thể là một trang bất kỳ. Địa chỉ trên trình duyệt của người dùng lúc này sẽ hiển thị đường dẫn của trang B. | | | | | | |
| 41 | | | JSP là gì? Nói: “Về bản chất JSP cũng chính là Servlet” đúng hay sai? Giải thích? So sánh giống và khác nhau giữa JSP và Servlet ? | |
| **JSP là viết tắt của JavaServer Pages**là một công nghệ để phát triển các trang web động. JSP giúp các nhà phát triển chèn java code vào các trang HTML bằng cách sử dụng các thẻ JSP đặc biệt,  hầu hết bắt đầu với <% và kết thúc với %>. | | | | | |
| |  |  |  | | --- | --- | --- | |  | **SERVLET** | **JSP** | | **GIỐNG** | **Đều là** công nghệ để phát triển các trang web động | | | **KHÁC** | **Được xây dựng dựa trên ngôn ngữ java** | **Được xây dựng dựa trên html.** | | Viết mã Servlet khó hơn JSP vì đó là HTML trong Java. | JSP rất dễ viết mã vì đó là Java trong HTML. | | * Trong mô hình MVC, **Servlet**sẽ xử lý phần controller. | **JSP** xử lý phần view. | |  | **JSP**chậm hơn **Servlet**do bước đầu tiên JSP thực hiện trong “vòng đời” là dịch JSP sang mã Java và biên dịch. | | **Servlet** có thể ghi đè phương thức service (), | **JSP**không thể ghi đè phương thức service | | | | | | |
| 42 | | | JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ? | |
| **JSP Standard Tag Library (JSTL) là một bộ thư viện thẻ được xây dựng dựa trên ngôn ngữ Java. Vì nếu viết như hiện tại, file JSP chứa cùng lúc 2 thành phần là HTML và Java, sẽ gây khó khăn cho việc phát triển => JSTL ra đời để hỗ trợ và khắc phục hạn chế này.**  Một số thẻ JSTL thông dụng:   |  |  | | --- | --- | | **Thẻ** | **Mô tả** | | <c:out> | Để viết một cái gì đó trong trang JSP, bạn có thể sử dụng EL cũng có thẻ này | | <c:import> | Giống với <jsp:include> hoặc chỉ thị include (include directive) | | <c:redirect> | Chuyển hướng (redirect) yêu cầu tới một nguồn dữ liệu khác. | | <c:set> | Sét đặt giá trị biến cho bởi phạm vi. | | <c:remove> | Loại bỏ biến ra khỏi phạm vi đã cho. | | <c:catch> | Bắt ngoại lệ và gói vào một đối tượng. | | <c:if> | Điều kiện logic đơn giản, sử dụng với EL và bạn có thể sử dụng nó để xử lý các trường hợp ngoại lệ từ *<c:catch>* | | <c:choose> | Tag có điều kiện đơn giản mà thiết lập một bối cảnh cho các trường hợp loại trừ lẫn nhau có điều kiện, đánh dấu bằng *<c:when>* và *<c:otherwise>* | | <c:when> | Thẻ con của *<c:choose>*, khi một điều kiện tại when là đúng. | | <c:otherwise> | Thẻ con của *<c:choose>*, khi tất cả các điều kiện *<c:when>* là sai. | | <c:forEach> | Dùng để lặp trên một tập hợp. | | <c:forTokens> | dùng để lặp trên chuỗi (tokens) được phân cách bởi một dấu phân cách. | | <c:param> | Được sử dụng với *<c:import>* để truyền các tham số. | | <c:url> | Để tạo một URL với các tham số tùy chọn (optional query string parameters). | | | | | | |
| 43 | | | Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC? | |
| Mô hình MVC là kiến trúc phần mềm phân thành 3 tầng:   * MODEL: là nơi xử lý nghiệp vụ, biểu diễn dữ liệu và tương tác với database (nếu có). * VIEW: là nơi hiển thị dữ liệu và là nơi tương tác với người dùng * CONTROLLER: là nơi tiếp nhận các request, xác định thành phần nào sẽ nhận nhiệm vụ xử lý yêu cầu nào và điều khiển luồng đi của chương trình. | | | | | |
| 44 | | | Sự khác nhau giữa MVC1 và MVC2 | |
| 45 | | | Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC? | |
| 46 | | | Phân biệt các loại Statement ? | |
| 47 | | | Kỹ thuật sử dụng transaction bằng JDBC là như thế nào? | |
| 48 | | | Các tính chất của Transaction? | |
| 49 | | | Phương pháp gọi 1 SP ở JDBC? | |
| 50 | | | Phân biệt Ant, Maven, Gradle? | |
| 43 | | | Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC? | |
|  | Mô hình MVC là kiến trúc phần mềm phân thành 3 tầng:   * MODEL: là nơi chứa các công cụ và tài nguyên của trang web, nhận nhiệm vụ xử lý nghiệp vụ, biểu diễn dữ liệu và tương tác với database (nếu có). * VIEW: là nơi hiển thị dữ liệu và là nơi tương tác với người dùng * CONTROLLER: là nơi tiếp nhận các request, xác định thành phần nào sẽ nhận nhiệm vụ xử lý yêu cầu nào và điều khiển luồng đi của chương trình. | | |
|  | 44 | Sự khác nhau giữa MVC1 và MVC2 | |
|  | |  |  | | --- | --- | | MVC1 | MVC2 | | 1. Trong MVC model 1: chỉ có 2 thành phần chính là trang JSP và file Bean, trong đó JSP đảm nhiệm cả View và Controller, file java Bean đảm nhiệm phần model. Cấu trúc:    1. Ví dụ về login, Khi A gửi request login từ form    2. Một trang jsp sẽ nhận được request    3. Trang Jsp sẽ sử dụng tài nguyên trong bean    4. Khi nhận được kết quả sẽ điều hướng đến trang JSP khác tùy theo kết quả nhân được    5. Hiển thị kết quả nhận được trong trong JSP tương ứng | 1. Trong MVC model 2 thì có thêm sự xuất hiện của servlet thay thế JSP trở thành controller. Cấu trúc:    1. A gửi request từ máy client đến server, controller sẽ nhận được request.    2. Servlet nhân dữ liệu từ request, gọi các tài nguyên từ Bean(Model).    3. Bean sẽ sử lí dữ liệu, gửi trả kết quả cho servlet    4. Servlet nhận kết quả trả về từ Model, sẽ điều hướng về view tương ứng với kết quả nhận được    5. Hiển thị trên view và trả lại máy client | |  |  | | | |
|  | 45 | Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC? | |
|  | 1. Nạp driver 2. Định nghĩa connection URL 3. Kết nối cơ sở dữ liệu bằng đối tượng connection đã tạo (=DriverManager.getConnection(link database, userName, password) 4. Tạo đối tượng statement 5. Thi hành câu truy vấn, nếu là câu SELECT (excuteQuery) thì dùng Resultset để hứng kết quả) 6. Xử lý kết quả 7. Đóng kết nối | | |
|  | 46 | Phân biệt các loại Statement ? | |
|  | |  |  |  | | --- | --- | --- | | STATEMENT | PREPARE STATEMENT | CALLABLE STATEMENT | | Hiệu năng thấp | Hiệu năng cao | | | Không có tham số | Có tham số | | | Không thể gọi | Không thể gọi | Có thể gọi store procedure | | Kém bảo mật | Bảo mật hơn | Kém bảo mật | | | |
|  | 47 | Kỹ thuật sử dụng transaction bằng JDBC là như thế nào? | |
|  | Là tạo ra một tiến trình xử lý có điểm đầu và điểm cuối, được chia ra thành nhiều hoạt động. Khi một tiến trình được thực hiện thì các hoạt động sẽ được thực hiện một cách tuần tự và độc lập. | | |
|  | 48 | Các tính chất của Transaction? | |
|  | Atomicity Atomicity quy định rằng tất cả các hoạt động của transaction hoặc là thực thi thành công hết hoặc là không có bất cứ hành động nào được thực khi có bất kỳ một hoạt động thực thi không thành công.  Vì vậy một transaction chỉ có 2 trạng thái:   * Abort: Tất cả các thay đổi của transaction sẽ không được đồng bộ xuống database. * Commit: Tất cả các thay đổi của transaction được đồng bộ xuống database.  Consistency Consistency nghĩa là tất cả các ràng buộc toàn vẹn dữ liệu(constraints, key, data types, Trigger, Check) phải được thực thi thành công cho mọi transaction phát sinh xuống database, nhầm đảm bảo tính đúng đắn của dữ liệu, nếu transaction gây ra những lỗi về ràng buộc dữ liệu, thì lúc này hệ thống sẽ không cho phép thực hiện và hủy transaction. | | |