**-­­CÂU HỎI AUDIT MODULE 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Q&A** | | | | | |
| 1 | **Q:** Repository là gì? Có bao nhiêu loại repository?  🡺 **A:**  - Là nơi chứa toàn bộ mã nguồn, bao gồm tất cả các file source code và lịch sử thay đổi từng file  - Có 2 loai repository:  + Remote repository: repo dùng chung thường được lưu trữ ở máy chủ (ex: Github)  + Local repository: repo lưu trên máy tính cá nhân của LTV | | | | | |
| 2 | **Q:** Các câu lệnh căn bản để làm việc với git?  🡺 **A:**  - git add: thêm thư mục cần đưa lên máy chủ  - git push: đưa thư mục lên máy chủ  - git commit: tạo snapshot (lát cắt)  - clone: sao chép mã nguồn giữa các repo | | | | | |
| 3 | **Q:** Giải thuật là gì? Các cách để biểu diễn thuật toán?  🡺 **A:**  - Repository bao gồm các chỉ thị (bước) để giải quyết mọi vấn đề  - Có 3 cách thông dụng để biểu diễn thuật toán:  + Mã giả (pseudo code)  + Lưu đồ (flowchart)  + NNLT | | | | | |
| 4 | **Q:** Mô tả thuật toán tìm kiếm 1 phần tử trong mảng bằng mã giả? | | | | | |
| 5 | **Q:** Mô tả thuật toán sắp xếp mảng số nguyên bằng mã giả? | | | | | |
| 6 | **Q:** Trình bày ý tưởng của 1 thuật toán bất kỳ? | | | | | |
| 7 | **Q:** Nêu các thể để tạo danh sách? Sự khác nhau giữa <ul> và <ol>  🡺 **A:**  - Các thẻ để tạo danh sách: <ul>, <li>, <ol>  - Sự khác nhau:  + <ul>: hiển thị kiểu chấm tròn đậm, ô vuông, hình tròn. Hiển thị không có thứ tự  + <ol>: hiển thị ra kèm danh sách đối tượng. Hiển thị có thứ tự | | | | | |
| 8 | **Q:** Các thẻ tạo nên 1 table? Phân biệt th và td?  🡺 **A:**  - Các thẻ tạo nên 1 table: <tr>, <td>, <th>  - Phân biệt th và td:  + th: khai báo 1 ô tiêu đề  + td: khai báo dữ liệu | | | | | |
| 9 | **Q:** Phân biệt innerHTML và innertext?  🡺 **A:**  - innertext và innerHTML đều có thể chèn text. Tuy nhiên innertext chỉ thuần chèn nội dung văn bản  - innerHTML chèn được thẻ HTML còn innertext thì không | | | | | |
| 10 | **Q:** Nêu 1 số thẻ HTML cơ bản mà bạn biết  🡺 **A:**  - Thẻ <h1 – h6>: dùng để hiển thị nội dung lên 1 trang web (h1 lớn nhất, h6 nhỏ nhất)  - Thẻ <p>: hiển thị nội dung  - Thẻ <br>: xuống dòng trong 1 đoạn văn bản  - Thẻ <center>: canh giữa nội dung văn bản  - Thẻ <hr>: kẻ 1 đường ngang trong văn bản  - Thẻ <nbsp>: tạo khoảng trống giữa 2 từ | | | | | |
| 11 | **Q:** Các thuộc tính cơ bản của thẻ form  🡺 **A:**  - Name: tên của form  - Action: mô tả URL của server, nơi mà dữ liệu được gửi đến  - Method: Xác định kiểm phương thức sử dụng để gửi dữ liệu, gồm có POST và GET | | | | | |
| 12 | **Q:** Phân biệt phương thức get và post trong thẻ form  🡺 **A:** | | | | | |
| Get | | | Post | | |
| - Dữ liệu hiển thị URL, không có bảo mật, có giới hạn dung lượng data khi gửi. (Thông thường được dùng cho màn hình  tìm kiếm, hiển thị chi tiết sản phẩm...) | | | - Dữ liệu không hiển thị trên URL, bảo mật hơn GET. Không giới hạn dung lượng data khi gửi. Thường được sử dụng với màn hình có dữ liệu nhạy cảm (mật khẩu, thông tin cá nhân của người dùng) | | |
| 13 | **Q:** Cách khai báo biến hằng từ khóa let và var khác nhau như thế nào? Phạm vi của biến  🡺 **A:** | | | | | |
| Let | | | Var | | |
| - Khai báo những giá trị có thể thay đổi  - Không cho phép khai báo lại biến với cùng 1 tên  - Không có cơ chế hoisting  - Phạm vi của let: nếu ở block scope thì vòng đời chỉ ở block scope | | | - Khai báo những giá trị có thể thay đổi  - Cho phép khai báo lại biến với cùng 1 tên  - Hoisting: tức là việc khai báo sẽ được đẩy lên đầu chương trình  - Phạm vi của var: nếu var khai báo ở block scope, thì vòng đời nằm ở ngoài block scope nữa | | |
| 14 | **Q:** Trong JS có bao nhiêu loại kiểu dữ liệu? Làm thế nào để xác đinh được biến có kiểu dữ liệu gì?  🡺 **A:**  - Có 2 kiểu dữ liệu:  + Kiểu nguyên thủy: Undefined, logic, kiểu số, kiểu chuỗi (string)  + Kiểu tham chiếu: object(ví dụ: null, [ ],), array  - Sử dụng type of để xác định kiểu dữ liệu | | | | | |
| 15 | **Q:** Các cách tạo chuỗi chứa dấu nháy?  🡺 **A:**  - C1: dùng dấu nháy khác với dấu nháy khai báo. Gõ dấu nháy đôi ở bên ngoài và dấu nháy đơn ở bên trong và ngược lại  - C2: dùng dấu / (/: bỏ qua) | | | | | |
| 16 | **Q:** NaN là gì? NaN === NaN có đúng không? Vì sao?  🡺 **A:**  - NaN: Not a number (không phải là số) là 1 giá trị kiểu number  - NaN !== NaN  - Vì NaN không phải là số, có thể là chuỗi, string, mảng thì không thể so sánh với vế còn lại được | | | | | |
| 17 | **Q:** Cho 1 bài toán liên quan đến ++ trước và ++ sau. Yêu câu dự đoán kết quả, và tại sao lại có kết quả như vậy?  🡺 **A:**  - ++ trước sẽ tăng giá trị lên 1 đơn vị rồi mới đánh giá biểu thức  - ++ sau sẽ đánh giá biển thức trước rồi mới tăng hoặc giảm giá trị 1 đơn vị | | | | | |
| 18 | **Q:** Các loại toán tử trong JS? Cho biết độ ưu tiên của các toán tử trong 1 biểu thức?  🡺 **A:**  - Toán tử số học: +, -, \*, /, %  - Toán tử gán: =, +=, ++, ...  - Toán tử so sánh: ==, ===, !=, !==, (>, <, <=, >=)  - Toán tử logic: && (và), || (hoặc), ! (phủ định)  - Độ ưu tiên: số học, so sánh, logic, gán. Dùng dấu ngoặc để thay đổi độ ưu tiên | | | | | |
| 19 | **Q:** Toán tử 3 ngôi là gì? Cú pháp?  🡺 **A:**  - Là một loại toán tử đặc biệt, nó gồm ba thành phần cấu thành biểu thức điều kiện  - Cú pháp: Biểu thức trả về true/false ? mệnh đề khi biểu thức trả về true : mệnh đề khi biểu thức trả về false | | | | | |
| 20 | **Q:** So sánh null và rỗng  🡺 **A:**  - Null: kiểu đối tượng, chưa cập phát bộ nhớ  - Rỗng: đã cấp phát bộ nhớ nhưng bộ nhớ có giá trị là rỗng, kiểu chuỗi (string) | | | | | |
| 21 | **Q:** Toán tử == và === khác và giống nhau như thế nào?  🡺 **A:**  - ==: chỉ so sánh giá trị  - ===: so sánh giá trị và kiểu dữ liệu | | | | | |
| 22 | **Q:** Phân biệt giữa 2 toán tử && và || trong JS?  🡺 **A:**  - &&: tức là tất cả biểu thức đều true => true  - ||: chỉ cần 1 biểu thức true => true | | | | | |
| 23 | **Q:** Các hàm dùng để tạo thông báo trong JS?  🡺 A:  - prompt: Nhận giá trị nhập từ người dùng  - alert: Hiển thị thông báo  - confirm: Cho người dùng lựa chọn Yes / No | | | | | |
| 24 | **Q:** Có bao nhiêu Statement control (câu lệnh điều khiển) trong JS?  🡺 A:  - Câu lệnh điều kiện, câu lệnh lặp, câu lệnh nhảy (return) | | | | | |
| 25 | **Q:** So sánh sự khác nhau giữa if và switch case?  🡺 A: | | | | | |
| if ... else if ... else | | | switch ... case | | |
| - So sánh >, <, =  - Biểu thức đầu vào là trả về true/false  - Không  - if ... else thay thế được switch ... case trong mọi trường hợp | | | - Chỉ so sánh = (so sánh ===)  - Biểu thức đầu vào là 1 biến string, number, logic  - Cần sử dụng break để thoát switch  - switch ... case chỉ thay thế cho if else trong trường hợp so sánh bằng | | |
| 26 | **Q:** Switch case so sánh == hay ===. Đặt ra trường hợp là so sánh bằng thì khi nào sử dụng if bậc thang? Khi nào sử dụng switch case?  🡺 A:  - switch ... case so sánh === (so sánh giá trị + kiểu dữ liệu)  - So sánh bằng khi biểu thức có ít hơn 3 điều kiện thì sử dụng if bậc thang  - Sử dụng switch ... case khi có nhiều hơn 3 điều kiện | | | | | |
| 27 | **Q:** Các biểu thức và luồng thực thi của for. Nếu thiếu 1 hoặc tất cả các biểu thức thì vòng for sẽ chạy như thế nào?  🡺**A:**  - Cú pháp:  for (initial – action(1); loop – condition (2); action – after – each – iteration (4) {  // body (3)  }  (1): Khối lệnh khởi tạo biến lặp, chỉ chạy 1 lần  (2): Khối lệnh điều kiện  (3): Nội dung cần lặp  (4): Khối lệnh tăng hoặc giảm biến lặp  - Luồng thực thi: (1) 🡺 (2) 🡺 (3) 🡺 (4) 🡺 (2) 🡺 (3) 🡺 (4). Lặp cho đến khi (2) không thỏa điều kiện thì thoát khỏi vòng lặp.  - Có thể thiếu khối lệnh điều kiện (khởi tạo ở ngoài chạy bình thường), hoặc khối lệnh tăng giảm (lặp vô tận)  - Có thể thiếu tất cả biểu thức (lặp vô tận) | | | | | |
| 28 | **Q:** Đặt ra 1 bài toán. Xác định bài toán cần sử dụng vòng lặp nào? | | | | | |
| 29 | **Q:** So sánh giống và khác nhau giữa for, while và do ... while  🡺**A:**  - Sử dụng vòng lặp for khi biết trước số lần lặp  - Sử dụng vòng lặp do ... while khi không biết trước số lần lặp. Lặp ít nhất 1 lần r mới xét điều kiện  - Sử dụng vòng lặp while khi hàm xét điều kiện rồi mới lặp | | | | | |
| 30 | **Q:** So sánh break và continue  🡺 **A:** | | | | | |
| Break | | | Continue | | |
| Dùng để thoát vòng lặp hiện tại | | | Dùng để nhảy đến vòng lặp tiếp theo | | |
| 31 | **Q:** Đặc điểm mảng 1 chiều trong JS  🡺 **A:**  - Là 1 biến đặc biệt (biến mảng), có thể lưu trữ được nhiều giá trị. Mỗi giá trị được gọi là 1 phần tử được lưu trữ ở các vị trí kế tiếp nhau trong ô nhớ.  - Chú ý:  + Quy tắc đặt tên mảng phải tuân theo quy tắc đặt tên biến  + Chỉ số (index) - vị trí của phần tử trong mảng, chỉ số bắt đầu từ 0 đến length – 1 (length: số lượng phần tử). | | | | | |
| 32 | **Q:** Các cách khởi tạo 1 mảng kiểu String trong JS  🡺 **A:**  - Dùng [ ] : let myArray = [“...”,”...”,”...”,...];  - Dùng new Array: let myArray = newArray (“...”.,” ...”, “...”, “...”);  - Dùng Array: let myArray = Array (“...”, “...”, “...”, “...”, “...”); | | | | | |
| 33 | **Q:** 1 số thao tác với mảng?  + Phân biệt push() và unshift()?  + Phân biệt push() và pop()?  + Phân biệt shift() và unshift()?  🡺 **A:** | | | | | |
| push() & unshift() | | push() & pop() | | shift() & unshift() | |
| - push(): thêm phần tử vào cuối mảng  - unshift(): thêm phần tử vào đầu mảng | | - push(): thêm phần tử vào cuối mảng  - pop(): xóa phần tử cuối | | - shift(): xóa phần tử đầu  - unshift(): thêm phần tử vào đầu mảng | |
| 34 | **Q:** Phân biệt tham trị và tham chiếu trong JS  🡺 **A:** | | | | | |
| Truyền tham trị (pass by value) | | | Truyền tham chiếu (pass by reference) | | |
| - Áp dụng cho kiểu nguyên thủy (số, bollean, ...)  - Giá trị của biến trước và sau khi gọi hàm function là không thay đổi | | | - Áp dụng cho kiểu đối tượng (object, array, ...)  - Giá trị của biến sau khi gọi function có thể bị thay đổi | | |
| 35 | **Q:** Phân biệt giữa hàm có return và hàm không có return  🡺 **A:** | | | | | |
| Hàm có return | | | Hàm không có return | | |
| - Dùng từ khóa “return” để trả về kết quả của 1 hàm  - Mỗi hàm chỉ có 1 giá trị trả về, các giá trị trả vể có thể là biến, số, chuỗi, mảng, đối tượng  - Gặp từ khóa return thì sẽ thoát khỏi hàm hiện tại  - Để nhận giá trị trả về của hàm thì sẽ dùng toán tử gán | | | - Hàm không có trả về | | |
| 36 | **Q:** Lập trình hướng đối tượng (Object Oriented Programming) là gì?  🡺 **A:**  - Là kỹ thuật lập trình cho phép LTV tạo ra các đối tượng trong source code, ánh xạ các đối tượng thực tế trong cuộc sống vào NNLT | | | | | |
| 37 | **Q:** Phân biệt class và object  🡺 A: | | | | | |
| Class | | | Object | | |
| - Dùng để mô tả nhóm các đối tượng có thuộc tính và hành vi tương tự nhau  - Ví dụ: class DienThoai, class Xe, ... | | | - Là thể hiện của class ( là sản phẩm được tạo ra từ class)  - Mỗi object có giá trị thuộc tính / hành vi khác nhau để phân biệt với các Object khác | | |
| 38 | **Q:** Các đặc điểm trong Lập trình hướng đối tượng. Đưa ra 1 ví dụ 1 trong 4 tính chất. (lưu ý cần phải trình bày được 4 tính chất bằng tiếng anh)  🡺 **A:** | | | | | |
| Tính bao đóng (Encapsulation) | Tính trừu tượng (Abstraction) | | Tính kế thừa (Inheritance) | | Tính đa hình (Polymorphism) |
| - Là khả năng truy xuất vào các thành phần của 1 đối tượng trong khi vẫn đảm bảo che giấu các đặc tính riêng tư bên trong đối tượng  - Triển khai tính bao đóng thì sẽ  dùng getter / setter | - Loại bỏ những thuộc tính và hành vi không quan trọng của đối tượng, chỉ giữ lại những thuộc tính và hành vi liên quan đến vấn đề đang giải quyết  - Loại bỏ sự phức tạp của đối tượng  - Ẩn đi các chi tiết triển khai, chỉ hiển thị các tính năng / kết quả đến với người dùng | | - Tính kế thừa cho phép các đối tượng có thể chia sẻ hay mở rộng các thuộc tính hoặc phương thức mà không cần tiến hành định nghĩa lại  - Kế thừa cho phép lớp con (child  class, subclass, derved class) sử dụng lại các đặc điểm và hành vi đã được định nghĩa trong lớp cha (super class, parent class, base class)  - Triển khai tính kế thừa thì dùng từ khóa extends | | - Là khả năng 1 đối tượng có thể hiện / hành vi theo nhiều cách khác nhau tùy thuộc vào ngữ cảnh  - Tính đa hình thể hiện khi với cùng 1  phương thức / hành vi nhưng có thể có cách ứng xử khác nhau ở những đối tượng khác nhau (ví dụ: phương thức : “nói”, ...) |
| 39 | **Q:** Constructor là gì? Trong 1 class có nhiều hơn 1 constructor được hay không?  🡺 **A:**  - Là hàm dùng để khởi tạo đối tượng  - Được gọi thông qua từ khóa new  - Trong JS chỉ có tối đa 1 constructor | | | | | |
| 40 | **Q:** Hàm nào cho phép tương tác với chuỗi ?  🡺 **A:**  - Tìm kiếm chuỗi con: indexOf(), lastIndexOf(), search()  - Cắt chuỗi con: slice(), substring(), substr()  - Chuyển hoa – thường: toUpperCase(), toLowerCase()  - Chuyển chuỗi sang mảng: split()  - Tìm kiếm kí tự trong chuỗi: charAt() | | | | | |
| 41 | **Q:** Các câu lệnh để vẽ hình tròn. Giải thích từng câu lệnh?  🡺 **A**:  - Khai báo HTML canvas để id width, height để khai báo vùng để vẽ  - Dùng script, lấy biến cavas và cho bằng document.getElementById  - Khai báo getContext 2D vẽ hình 2 chiều  - Để vẽ khai báo beginPath để bắt đầu  - Gọi hàm arc (tọa độ x, tọa độ y, bán kính,0 (góc bắt đầu), góc kết thúc (2PI (vẽ 1 hình tròn), PI(vẽ nửa đường tròn dưới), (PI – 0: nửa đường tròn trên))  - lineWidth: để chỉnh nét vẽ  - strokeStyle: để chỉnh màu  - stroke(): để vẽ | | | | | |

**CÂU HỎI AUDIT MODULE 2**

|  |  |
| --- | --- |
| **STT** | **Q&A** |
| 1 | **Q:** JAVA là gì ?  🡺 **A:**  - Là 1 NNLT hướng đối tượng (OOP: Object Oriented Programming) được phát triển bởi Sun Microsystem do Jame Gosling khởi xướng (hiện tại của Oracle)  - Write once, run anywhere |
| 2 | **Q:** JDK, JRE, JVM  🡺 **A:**  - JDK: Java Development Kit: bộ công cụ lập trình Java  - JRE: Java Runtime Enviroment: môi trường thực thi Java  - JVM: Java Virtual Machine: máy ảo java 🡺 đối với mỗi hệ điều hành thì sẽ được cài đặt 1 máy ảo riêng phù hợp với mỗi máy tính trên hệ điều hành đó 🡺 viết 1 lần và chạy mọi nơi |
| 3 | **Q:** Thông dịch, biên dịch khác nhau như thế nào ?  🡺 **A:**   |  |  | | --- | --- | | Biên dịch | Thông dịch | | - Sẽ chuyển từ file java (.java) 🡺 mã byte code (.class). Dịch toàn bộ file 1 lần và sau đó sử dụng kết quả đó để thông dịch. Và nếu có cập nhập file java thì chương trình sẽ cập nhập lại file byte code mà không cần phải biên dịch toàn bộ file 1 lần nữa | - Quá trình dịch từng dòng lệnh, muốn chạy lại phải dịch lại 1 lần nữa | |
| 4 | **Q:** JAVA là ngôn ngữ thông dịch hay biên dịch ?  🡺 **A:**  - Java là 1 ngôn ngữ vừa biên dịch, vừa thông dịch |
| 5 | **Q:** Java Platform là gì?  🡺 **A:**  - Là một tập hợp các chương trình giúp phát triển và chạy các chương trình được viết bằng ngôn ngữ lập trình Java.  - **Java Platform** bao gồm một công cụ thực thi (execution engine), trình biên dịch (compiler) và một bộ thư viện Java. |
| 6 | **Q:** Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.  🡺 **A:**  - Có 2 loại kiểu dữ liệu:  + Kiểu nguyên thủy: 8 kiểu   * Số nguyên: byte(1 byte), short(2 byte), int(4 byte), long(8 byte) * Số thực: float(4 byte), double(8 byte) * Kiểu ký tự: char(2 byte) * Boolean (1 bit)   + Kiểu đối tượng: string, date, math, object, ... |
| 7 | **Q:** Sự khác nhau giữa bộ nhớ heap và stack trong Java?  🡺 **A:**  - Trong C++:   |  |  | | --- | --- | | Bộ nhớ heap | Bộ nhớ stack | | - Được dùng để lưu trữ các biến toàn cục (global variable) của chương trình, hoặc lưu trữ các đối tượng của con trỏ khi được cấp phát. Vùng nhớ Heap phải được giải phóng thông qua hàm bởi lập trình viên | - Được dùng để lưu trữ các biến cục bộ (local variable) bên trong hàm, vùng nhớ này dùng để chứa giá trị của các tham số khi được gọi đến. Khi kết thúc một hàm, vùng nhớ Stack sẽ được tự động giải phóng. Stack sử dụng theo cấu trúc **LIFO** |   - Trong Java:   |  |  | | --- | --- | | Bộ nhớ heap | Bộ nhớ stack | | - Được sử dụng để lưu trữ các đối tượng trong Java | - Được sử dụng để lưu trữ biến cục bộ và gọi hàm | |
| 8 | **Q:** Trình bày các cách khởi tạo một mảng trong JAVA ?  🡺 **A:**  - Có 2 cách khởi tạo mảng:  + nameArray = {value 1, value 2, ..., value n}  + nameArray = new ElementType [length]; |
| 9 | **Q:** Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?  🡺 **A:**  - Phần tử của mảng có thể dùng kiểu dữ liệu: int, double, String, ...  - Giá trị mặc định là null |
| 10 | **Q:** OOP là gì ?  🡺 **A:**  - Object Oriented Programming là 1 kỹ thuật lập trình cho phép ánh xạ các đối tượng trong thực tế vào lập trình |
| 11 | **Q:** Phân biệt class và object  🡺 **A:**   |  |  | | --- | --- | | Object | Class | | Là thể hiện của 1 class | Là khuôn mẫu định nghĩa ra các thuộc tính và hành vi chung của tất cả các đối tượng cùng loại | | Đối tượng được khởi tạo bằng từ khóa new | Sử dụng từ khóa “class” để khai báo class | | Đối tượng có thể được tạo nhiều | Class (có cùng tên) chỉ khai báo 1 lần (trong cùng package) | |
| 12 | **Q:** Constructor là gì ?  🡺 **A:**  - Là 1 phương thức đặc biệt để khởi tạo object |
| 13 | **Q:** Cách khai báo constructor và đặc điểm constructor trong JAVA  🡺 **A:**  - Cách khai báo constructor:  Private nameClass(thuộc tính) (không có kiểu trả về) {  This. thuộc tính = thuộc tính;  ......  // body  }  - Đặc điểm:  + Nếu không khai báo constructor thì Java sẽ cung cấp 1 constructor mặc định không tham số  + Tên của constructor phải trùng với tên class  + Không có kiểu trả vể  + Trong 1 class có thể có nhiều constructor  + Có thể gọi constructor bằng từ khóa this. Lưu ý: lệnh gọi phải được để trên cùng của body và chỉ có thể gọi duy nhất 1 constructor |
| 14 | **Q:** Phân biệt constructor và method  🡺 **A:**   |  |  | | --- | --- | | Method | Constructor | | - Là 1 nhóm các câu lệnh để thực hiện 1 nhiệm vụ nào đó của đối tượng  - Có kiểu trả về  - Method không được tạo ra bởi trình biên dịch Java  - Tên method có thể giống hoặc khác tên lớp | - Là 1 phương thức đặc biệt để khởi tạo object  - Không có kiểu trả về  - Java cung cấp mặc định 1 constructor không tham số  - Tên constructor phải trùng với tên class | |
| 15 | **Q:** Tính bao đóng là gì ? Làm sao để thu được tính bao đóng trong java ?  🡺 **A:**  - Là 1 kỹ thuật ẩn giấu thông tin của đối tượng, chỉ hiển thị những thông tin cần thiết  - Để đạt được tính bao đóng trong Java thì sử dụng:  + Access modifier (khai báo biến của 1 lớp là private)  + Getter và setter (sửa đổi và xem các giá trị của biến) |
| 16 | **Q:** Tham trị, tham chiếu  🡺 **A:**   |  |  | | --- | --- | | Tham trị (truyền giá trị) | Tham chiếu (chiếu đến 1 địa chỉ) | | - Dành cho các biến, các tham số khai báo kiểu nguyên thủy  - Gọi một phương thức và truyền một giá trị cho phương thức đó được gọi là truyền giá trị. Việc thay đổi giá trị chỉ có hiệu lực trong phương thức được gọi, không có hiệu lực bên ngoài phương thức. Vì nó chỉ tác động đến bản sao giá trị chứ không phải là địa chỉ nó trong bộ nhớ | - Địa chỉ đối tượng được lưu trong bộ nhớ Heap nhưng biến tham chiếu được lưu trong bộ nhớ Stack  - Gọi một phương thức và truyền một tham chiếu cho phương thức đó được gọi là truyền tham chiếu. Việc thay đổi giá trị của biến tham chiếu bên trong phương thức làm thay đổi giá trị gốc của nó. | |
| 17 | **Q:** Từ khóa static dùng để làm gì ?  🡺 **A:**  - Dùng để khai báo thuộc tính, phương thức, khối (block) của class, không phải của đối tượng  - Mục đích sử dụng:  + Định nghĩa các thuộc tính, phương thức dùng chung cho toàn bộ đối tượng  + Tạo ra các lớp tiện ích (lớp tiện ích là lớp mà toàn bộ thuộc tính và phương thức là static) |
| 18 | **Q:** Ràng buộc khi sử dụng static  🡺 **A:**  - Phương thức static chỉ có thể gọi phương thức static khác  - Phương thức static chỉ có thể truy cập các biến static khác  - Chỉ khởi tạo giá trị biến static thông qua khối static hoặc gán trực tiếp khi khai báo |
| 19 | **Q:** Các loại biến trong JAVA  🡺 **A:**  - Biến cục bộ/local: khai báo bên trong constructor, method và {}  - Biến instance: khai báo bên trong class, bên ngoài constructor, method và block scope  - Biến static: được khai báo bằng từ khóa static |
| 20 | **Q:** Trình bày các loại access modifier, và phạm vi truy cập.  🡺 **A:**  - Là những từ khóa quy định phạm vi truy cập của biến, method 3 từ khóa và 4 level   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Access modifier | Trong lớp | Trong package | Ngoài package nhưng phải bởi lớp con | Ngoài package | | Private | X |  |  |  | | Default (X) | X | X |  |  | | Protected | X | X | X |  | | Public | X | X | X | X | |
| 21 | **Q:** Kế thừa trong JAVA là gì ?  🡺 **A:**  - Là cơ chế cho phép 1 lớp con sử dụng lại các đặc điểm và hành vi của lớp cha  - Mục đích là tái sử dụng source code  - Mối quan hệ giữa lớp con và cha là mối quan hệ is - a |
| 22 | **Q:** Lớp con kế thừa được những tài sản nào (thuộc tính, phương thức) của lớp cha ?  🡺 **A:**  - Kế thừa được tất cả các thuộc tính, phương thức của lớp cha  - Không thể truy cập vào private của lớp cha  - Không kế thừa constructor của lớp cha  - Java không hỗ trợ đa kế thừa |
| 23 | **Q:** Lớp Object là gì?  🡺 **A:**  - Là lớp gốc của tất cả các lớp trong Java  - toString() là phương thức return về chuỗi biểu diễn của đối tượng |
| 24 | **Q:** Khái niệm đa hình  🡺 **A:**  - Là khả năng mà 1 đối tượng có thể hiện/hành vi theo nhiều cách khác nhau tùy thuộc vào ngữ cảnh |
| 25 | **Q:** Phân biệt Overloading (ghi đè) và Overriding (nạp chồng)  🡺 **A:**   |  |  | | --- | --- | | Overriding (ghi đè) | Overloading (nạp chồng) | | - Là cơ chế cho phép lớp con định nghĩa lại phương thức đã có trước ở lớp cha | - Là cơ chế cho phép 1 lớp có khả năng định nghĩa ra nhiều phương thức | | - Cùng tên, cùng danh sách tham số, cùng kiểu trả về | - Có cùng tên nhưng khác nhau về danh sách tham số | | - Access modifier của lớp con phải có level bằng hoặc lớn hơn ở lớp cha |  | | - Xảy ra trong class có mối quan hệ is – a (kế thừa) |  | |
| 26 | **Q:** Ép kiểu là gì ? các loại ép kiểu  🡺 **A:**  - Ép kiểu là việc gán giá trị của một biến có kiểu dữ liệu này tới biến khác có kiểu dữ liệu khác  - Có 2 loại ép kiểu:  + Ép kiểu ngầm đinh: diễn ra 1 cách tự động bởi hệ thống  + Ép kiểu tường minh: không thể ép tự động được mà coder phải ghi rõ kiểu dữ liệu cần ép về qua cú pháp: (kiểu dữ liệu) tên\_biến |
| 27 | **Q:** Tính trừu tượng là gì ?  🡺 **A:**  - Là khả năng ẩn đi các chi tiết của trình triển khai, chỉ hiển thị các tính năng/kết quả cho người dùng. Chỉ quan tâm đến kết quả đạt được, không quan tâm đến cách thực hiện. |
| 28 | **Q:** Phân biệt abstract class và interface  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | Abstract class | Interface | | Định nghĩa | - Là class nhưng có tính trừu tượng cao | - Là 1 bản thiết kế của 1 lớp, quy định hành vi chung cho lớp triển khai nó (chứa các phương thức trừu tượng – phương thức không có cài đặt) | | Tính chất | - Không thể tạo được đối tượng | - Không thể khởi tạo đối tượng interface | | - 1 lớp chứa abstract method thì bắt buộc lớp đó phải là abstract class (ngược lại không) | - Các method toàn bộ là abstract method | | - Lớp chứa abstract có thể chứa thuộc tính và phương thức bình thường | - Các trường thuộc tính của interface đều là static final | | - Không dùng từ khóa final cho lớp abstract | - Không thể khởi tạo constructor | | - Lớp abstract có thể kế thừa 1 lớp abstract khác (không cần override lại phương thức abstract) | - Có thể extend từ nhiều interface khác | |
| 29 | **Q:** So sánh Array và ArrayList  🡺 **A:**   |  |  | | --- | --- | | Array | ArrayList | | - Có kích thước cố định | - Kích thước thay đổi được | | - Có thể lưu được kiểu dữ liệu nguyên thủy | - Lưu được kiểu đối tượng. Nếu muốn lưu kiểu nguyên thủy thì phải sử dụng wrapper class với cơ chế auto boxing | | - Chỉ có thuộc tính length | - Có nhiều hàm hỗ trợ | | - Tốc độ lưu trữ và thao tác nhanh | - Tốc độ lưu trữ và thao tác chậm | |
| 30 | **Q:** So sánh ArrayList và LinkedList  🡺 **A:**   |  |  | | --- | --- | | ArrayList | LinkedList | | - Sử dụng mảng để lưu trữ | - Danh sách liên kết | | - Truy xuất ngẫu nhiên nhanh | - Truy xuất ngẫu nhiên chậm | | - Thêm và xóa chậm | - Thêm và xóa nhanh | |
| 31 | **Q:** So sánh Set, List và Queue ?  🡺 **A:**   |  |  |  | | --- | --- | --- | | Set | List | Queue (hàng đợi) | | - Lưu trữ các phần tử không trùng lặp | - Là cấu trúc dữ liệu tuyến tính, có thể xác định thứ tự của các phần tử. Cho phép các phần tử được trùng lặp | - Hoạt động theo cơ chế FIFO (first in – first out) | |
| 32 | **Q:** Set là gì, các lớp triển khai của Set  🡺 **A:**  - Set là 1 Collection không thể chứa các phần tử trùng lặp  - Các lớp triển khai của Set:  + HashSet: các phần tử được lưu dưới dạng mã băm và không duy trì thứ tự chèn vào  + LinkedHashSet: các phần tử được lưu dưới dạng mã băm nhưng duy trì thứ tự chèn vào  + TreeSet: các phần tử sắp xếp tăng dần  + EnumSet |
| 33 | **Q:** Generic là gì ?  🡺 **A:**  - Là cơ chế cho phép sử dụng kiểu dữ liệu như là tham số (tham số hóa kiểu dữ liệu). Cho phép chúng ta tạo ra các class, method, interface hoạt động với nhiều kiểu dữ liệu khác nhau 🡺 Tái sử dụng mã nguồn 🡺 Tiết kiệm chi phí phát triển |
| 34 | **Q:** Ưu điểm và nhược điểm khi dùng generic ?  🡺 **A:**  - Ưu điểm:  + Không cần ép kiểu  + Phát hiện lỗi ngay tại compile time  + Tạo ra các lớp tổng quát có thể áp dụng cho nhiều kiểu dữ liệu khác nhau  - Nhược điểm:  + Không thể generic kiểu nguyên thủy  + Không thể tạo instance của 1 kiểu dữ liệu generic  + Không thể ép kiểu tường minh generic |
| 35 | **Q:** Stack là gì, các phương thức của stack ?  🡺 **A:**  - Là 1 cấu trúc dữ liệu dạng danh sách, thêm và lấy các phần tử theo nguyên tắc FILO (first in – last out)  - 1 số phương thức hỗ trợ của stack:  + push (thêm phần tử 🡺 thêm vào cuối)  + peek (lấy ra xem nhưng không xóa phần tử)  + pop(): lấy ra xem và xóa phần tử khỏi stack  + isEmpty(): kiểm tra stack có bị rỗng hay không  + size(): trả về số lượng phần tử  + ... |
| 36 | **Q:** Queue là gì, các class triển khai của queue ?  🡺 **A:**  - Là hàng đợi là 1 cấu trúc dữ liệu dạng danh sách, thêm và lấy phần tử theo quy tắc FIFO (first in – first out)  - Các lớp triển khai của Queue  + LinkedList  + ArrayDeque  + PriortyQueue |
| 37 | **Q:** Phương thức cơ bản của queue ?  🡺 **A:**  - Thêm phần tử:  + add(): thành công là true, thất bại là lỗi Exception  + offer: thành công là true, thất bại là false  - Xóa phần tử:  + remove(): nếu stack đang rỗng thì ném ra Exception  + poll(): nếu stack đang rỗng thì trả về null  - Lấy ra xem:  + element: nếu stack đang rỗng thì ném ra Exception  + peek: nếu stack đang rỗng thì trả về null |
| 38 | **Q:** So sánh Comparable và Comparator, khi nào dùng cái nào ?  🡺 **A:**  - So sánh Comparable và Comparator   |  |  | | --- | --- | | Comparable | Comparator | | - Nó cung cấp phương thức compareTo() | - Nó cung cấp phương thức compare() | | - Nó được đặt trong java.lang package | - Nó được đặt trong java.util package | | - 1 lớp được implement Comparable interface, thì lớp đó phải được sửa đổi | - Lớp không bị sửa đổi | | - Sắp xếp các phần tử bởi phương thức Collections.sort(List) | - Sắp xếp các phần tử bởi phương thức Collections.sort(List, Comparator) |   - Khi nào sử dụng:  + Phương thức compareTo() chỉ được overide lại 1 lần và chỉ có thể xét được 1 tiêu chí nhất định  + Phương thức compare() có thể được gọi nhiều lần và định nghĩa ngay tại vị trí cần so sánh để sắp xếp theo các tiêu chí tùy ý |
| 39 | **Q:** Map là gì, các class triển khai  🡺 **A:**  - Map sử dụng để lưu trữ và truy xuất theo cặp khóa key và value  - Các class để triển khai:  + HashMap  + LinkedHashMap  + TreeMap |
| 40 | **Q:** Phương thức equals() và hashcode() trong Java là gì?  🡺 **A:**   |  |  | | --- | --- | | Equals() | Hashcode() | | - Phương thức equals được dùng cho việc tìm kiếm phần tử đã tồn tại trong tập hợp hay không và nó cũng được dùng để xoá một phần tử ra khỏi tập hợp. | - hashCode có nghĩa là sẽ trả về một số duy nhất tương ứng với từng object. Mỗi Object trong java đều có 1 định danh và định danh để phân biệt mỗi object chính là giá trị hashcode. | |
| 41 | **Q:** Cây nhị phân là gì ?  🡺 **A:**  - Là cây chỉ có 0, 1 hoặc 2 node con tương ứng với mỗi node cha |
| 42 | **Q:** Trình bày các cách duyệt cây nhị phân.  🡺 **A:**  - Có 4 cách duyệt:  + Pre-order: duyệt nút gốc đầu tiên, sau đó sẽ duyệt cây con bên trái và cuối cùng sẽ duyệt cây con bên phải  + In-order: cây con bên trái được duyệt đầu tiên, sau đó là nút gốc và tiếp theo là cây con bên phải.  + Post-order: duyệt cây con bên trái, sau đó sẽ duyệt cây con bên phải và cuối cùng là duyệt nút gốc.  + Breath-first: duyệt nút gốc đầu tiên, sau đó sẽ duyệt lần lượt các cây con từ trái sang phải |
| 43 | **Q:** Ngoại lệ (Exception là gì) ?  🡺 **A:**  - Là 1 sự kiện bất thường xảy ra trong quá trình thực thi chương trình, nó phá vỡ luồng thực thi bình thường của chương trình, thậm chí có thể gây chết chương trình |
| 44 | **Q:** Phân loại Exception  🡺 **A:**  - Có 2 loại:  + CheckedException: xảy ra trong quá trình compile time, không thể bỏ qua trong quá trình compile, bắt buộc phải handle (xử lý nó). Ví dụ: FileNotFoundException, DatabaseException,...  + UncheckedException: xảy ra trong quá trình runtime, lỗi này không chắc chắn xảy ra. Loại này có thể bỏ qua trong quá trình compile, không bắt buộc phải handle (xử lý nó). Ví dụ: NumberFormatException, ArrayIndexOutOfBoundException,... |
| 45 | **Q:** Phân biệt Error và Exception  🡺 **A:**   |  |  | | --- | --- | | Error | Exception | | - Xảy ra khi hệ thống tài nguyên bị thiếu  - Không bao giờ có thể phục hồi  - Không bao giờ có thể được xử lý  - Được định nghĩa trong gói java.lang.Error | - Xảy ra nếu mã có vấn đề  - Có thể được phục hồi bằng cách chuẩn bị mã để xử lý ngoại lệ  - Có thể xử lý  - Được xác định trong gói java.lang.Exception | |
| 46 | **Q:** Có bao nhiêu cách để xử lý ngoại lệ (handle)  🡺 **A:**  - Có 2 cách:  + try – catch: để xử lý ngay tại đoạn mã bị lội  + throws: dùng để ném ngoại lệ cho phương thức khác xử lý |
| 47 | **Q:** Một số lưu ý khi dùng try-catch  🡺 **A:**  - Các khối catch phải được sắp xếp từ cụ thể đến chung nhất (từ exception con đến exception cha)  - Trong cùng 1 thời điểm chỉ có 1 khối catch được thực thi  - Khối finally sẽ luôn được thực thi dù chương trình có xảy ra ngoại lệ hay không  - Khối try có thể không có hoặc có nhiều khối catch nhưng chỉ có 1 khối finally mà thôi |
| 48 | **Q:** Serialization là gì ? lưu ý khí sử dụng Serialization  🡺 **A:**  - Là cơ chế để chuyển đổi trạng thái của 1 đối tượng vào 1 chuỗi byte stream. Ngược lại, chuyển đổi từ byte stream sang object được gọi là deserialization  - Lưu ý:  + Với 2 class có quan hệ kế thừa thì class cha cần triển khai interface serialization và class con không cần triển khai lại  + Thuộc tính của đối tượng phải implement serialization |

**CÂU HỎI AUDIT MODULE 3**

|  |  |
| --- | --- |
| STT | **Q&A** |
| 1 | **Q:** Cơ sở dữ liệu (database) là gì ?  🡺 **A:** Là một hệ thống thông tin có cấu trúc được lưu trữ trên các hệ quản trị cơ sử dữ liệu nhằm thỏa mãn nhu 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 | **Q:** Phân biệt các câu lệnh DML & DDL? DML, DDL bao gồm các câu lệnh nào?  🡺 **A:**   |  |  | | --- | --- | | DDL | DML | | - Nhóm các câu lệnh liên quan đến việc định nghĩa dữ liệu  - Các lệnh: Create, Alter, Drop | - Nhóm các câu lệnh liên quan tới thao tác dữ liệu  - Các lệnh: Insert (into), Update (...set), Delete | |
| 3 | **Q:** 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?  🡺 **A:**  - Mô hình thực thể mối quan hệ là 1 sơ đồ hiển thị mối quan hệ của các tập thực thể lưu trữ trong CSDL, giúp giải thích cấu trúc logic của CSDL  - Các thành phần: Thực thể và tập thực thể, thuộc tính, mối quan hệ giữa các tập thực thể, tập mối quan hệ |
| 4 | **Q:** Các bước chuyển đổi từ ERD 🡺 Table?  🡺 **A:**  - Biến đổi thực thể → table  + Thuộc tính khoá của thực thể là khoá chính của bảng  + Thuộc tính thường có của thực thể là thuộc tính thường của bảng  - Biến đổi thuộc tính đa trị  + Tách thuộc tính đa trị ra bảng mới  + Đặt khoá ngoại vào bảng hiện tại  - Biến đổi 1-1  + Đặt thuộc tính khoá ở phía tuỳ chọn → phía bắt buộc  - Biến đổi 1-n  + Đặt thuộc tính khoá ở 1→n  - Biến đổi n-n  + Tạo ra một bảng mới có khoá chính là khoá phức hợp 2 khoá chính của 2 thực thể tham gia vào mqh → khoá composite |
| 5 | **Q:** Các bước chuẩn hóa dữ liệu?  🡺 **A:** Có 3 dạng chuẩn hóa:  - Dạng chuẩn hóa 1NF: các thuộc tính phải là đơn trị, tách các thuộc tính đa trị thành bảng mớim hhjkm n k……,  - Dạng chuẩn 2NF: đạt dạng chuẩn 1NF và tất cả các thuộc tính không khoá đều phụ thuộc đầy đủ vào mọi khoá  - Dạng chuẩn 3NF: đạt chuẩn 2NF và tất cả các thuộc tính không khóa phải phụ thuộc bắc cầu vào thuộc tính khóa |
| 6 | **Q:** Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL?  🡺 **A:**  - Ràng buộc (constraint) là các quy tắc được quy định cho bảng  - Các loại ràng buộc trong MySQL:  + NOT NULL: Không cho phép giá trị null  + UNIQUE: Mỗi giá trị là duy nhất, không cho phép trùng lặp  + PRIMARY KEY: Khóa chính, không được null và không được trùng lặp  + FOREIGN KEY: Khóa ngoại, được phép null  + CHECK: Kiểm tra giá trị theo điều kiện nào đó  + DEFAULT: Giá trị mặc định của cột khi không nhập dữ liệu  + INDEX: Tăng tốc độ truy vấn |
| 7 | **Q:** Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?  🡺 **A:**  - Khóa chính (Primary key) là một cột (hoặc nhiều cột) được sử dụng để xác định 1 bản ghi duy nhất trong bảng. Mỗi bảng chỉ có thể có 1 khóa chính  - |
| 8 | **Q:** Khóa ngoại (foreign key) là gì?  🡺 **A:**  - Khóa ngoại là cơ chế để tạo liên kết giữa 2 bảng trong cùng CSDL  - Khóa ngoại được đặt trên 1 cột của bảng này và tham chiếu đến khóa chính của 1 bảng khác  - Kiểu dữ liệu của khóa chính và khóa ngoại phải giống nhau |
| 9 | **Q:** Phân biệt DELETE và TRUNCATE?  🡺 **A:**   |  |  | | --- | --- | | DELETE | TRUNCATE | | - Cú pháp: DELETE FROM table\_name WHERE conditions;  - Nếu muốn xóa cả bảng: DELETE FROM table\_name;  - Khi chạy lệnh DELETE thì SQL sẽ log lại từng dòng đã xóa vào transaction log, vì thế nên khi tạo 1 record mới, giá trị của id sẽ không bắt đầu từ 1 mà sẽ có giá trị n+1 với n là giá trị của record cuối cùng được tạo.  - Xoá bằng delete thì sẽ restored được  - Chậm hơn truncate | - Cú pháp: TRUNCATE TABLE table\_name;  - Đối với TRUNCATE chỉ có thể dùng cho bảng chứ không thể dùng cho từng record.  - Khi chạy lệnh TRUNCATE thì SQL sẽ xóa hết dữ liệu của bảng và reset transaction log, vì thế khi tạo 1 record mới, giá trị của id sẽ bắt đầu từ 1  - Xoá bằng truncate thì không restored được  - Nhanh hơn vì nó tiêu tốn ít bộ nhớ hơn | |
| ưii10 | **Q:** Các loại JOIN? Phân biệt?  🡺 **A:**  - Inner join: Nối bảng lấy những record chung giữa 2 bảng  - Outer join: Nối bảng lấy phần tử chung và phần tử riêng  + Left outer join (left join): phần tử chung + phân tử riêng của bảng bên trái  + Right outer join (right join): phần tử chung + phần tử riêng của bảng bên phải  - Cross join: 1 phần tử của bảng này sẽ nối với tất cả các phần tử của bảng kia  - Self join: Tự nối với chính mình, trong phép self join có thể có inner join, outer join hoặc cross join  - Excluding join: Join 2 bảng lại với nhau và lấy phần tử riêng  + Left excluding join: left join loại đi các phần tử chung chỉ lấy phần tử riêng của bảng bên trái  + Right excluding join: right join loại đi các phần tử chung chỉ lấy phần tử riêng của bảng bên phải  + Full excluding join: full chỉ lấy phần tử riêng của 2 bảng bên trái và bên phải |
| 11 | **Q:** Phân biệt JOIN và UNION?  🡺 **A:**   |  |  | | --- | --- | | JOIN | UNION | | - Cho phép kết nối dữ liệu liên quan trong các bảng khác nhau | - Hợp dữ liệu của 2 bảng mà có cột loại data giống nhau | |
| 12 | **Q:** Phân biệt UNION và UNION ALL?  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | UNION | UNION ALL | | Trả lại giá trị | - Lệnh union trả về các giá trị khác biệt với nhau | - Hiển thị tất cả các giá trị cụ thể cho truy vấn, bao gồm các giá trị đang được sao chép | | Loại bỏ sự trùng lặp | - Luôn loại bỏ các giá trị trùng lặp và trình bày các giá trị riêng | - Không loại bỏ các giá trị trùng lặp | | Tốc độ | - Chậm hơn | - Tương đối nhanh | | Đề xuất | - Các giá trị cần được sắp xếp và thống nhất | - Các giá trị cần phải được thống nhất, nhưng không cần phải sắp xếp | |
| 13 | **Q:** Phân biệt EXIST và IN?  🡺 **A:**   |  |  | | --- | --- | | EXIST | IN | | - Kiểm tra một giá trị có xuất trong tập giá trị của truy vấn con hay không  - Quét từng bộ ở câu truy vấn con và so sánh trực tiếp với yêu cầu mẹ, nếu không thỏa sẽ loại trừ ngay | - Kiểm tra xem một truy vấn con có trả về kết quả nào không, trả về true nếu có ít nhất 1 dòng và trả về false nếu không có dòng nào  - Sẽ tìm ra tất cả các bộ thỏa yêu cầu truy vấn con rồi mới so sánh với câu truy vấn mẹ | |
| 14 | **Q:** Phân biệt AND và OR?  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | AND | OR | | Cú pháp | WHERE [DIEU\_KIEN1] AND [DIEU\_KIEN2] ... AND [DIEU\_KIENN] | WHERE [DIEU\_KIEN1] OR [DIEU\_KIEN2] ... OR [DIEU\_KIENN] | | Dữ liệu lấy ra | - Phải thỏa mãn cùng lúc tất cả các điều kiện có trong mệnh đề WHERE | - Chỉ cần thỏa mãn 1 trong các điều kiện được liệt kê | |
| 15 | **Q:** Phân biệt GROUP BY và ORDER BY?  🡺 **A:**   |  |  | | --- | --- | | GROUP BY | ORDER BY | | - Để sắp xếp dữ liệu giống nhau thành các nhóm  - Mệnh đề GROUP BY tuân theo mệnh đề WHERE trong câu lệnh SELECT và đứng trước mệnh đề ORDER BY | - Được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần, dựa trên một hoặc nhiều cột  - Lệnh ASC được sử dụng để sắp xếp tăng dần và DESC được sử dụng để sắp xếp giảm dần | |
| 16 | **Q:** Muốn nối chuỗi trong MySQL thì sử dụng gì?  🡺 **A:** Sử dụng CONCAT()  - Cú pháp: CONCAT(str1, str2, ..., strn) 🡪 Nối chuỗi |
| 17 | **Q:** Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?  🡺 **A:**  SELECT AVG(Diem) FROM SINHVIEN |
| 18 | **Q:** Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào?  🡺 **A:**  SELECT id, name, AVG(Diem) as Diem  FROM SINHVIEN; |
| 19 | **Q:** 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?  🡺 **A:**  SELECT id, name, AVG(Diem) as Diem  FROM SINHVIEN  GROUPBY id, name;  HAVING AVG(Diem) > 5; |
| 20 | **Q:** Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID)?  🡺 **A:**  SELECT studentId, studentName, className  FROM STUDENT s  JOIN CLASS c ON s.classId = c.classId; |
| 21 | **Q:** Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?  🡺 **A:**  SELECT \* FROM SINHVIEN ORDER BY DIEMTHI  (tăng: asc, giảm: desc) |
| 22 | **Q:** 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?  🡺 **A:**  SELECT id, name, age FROM SINHVIEN  ORDER BY name DESC, age; |
| 23 | **Q:** Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?  🡺 **A:**  SELECT id, ho, ten FROM SINHVIEN  WHERE ho = “Le” and ten = “Hai” |
| 24 | **Q:** Hiển thị các sinh viên có tên bắt đầu là ‘C’?  🡺 **A:**  SELECT id, name FROM SINHVIEN  Where name like “C%” |
| 25 | **Q:** Làm sao để có thể tăng tốc độ truy vấn? Nhược điểm là gì?  🡺 **A:**  - Dùng INDEX để tăng tốc độ truy vấn  - Nhược điểm: thêm, xóa, sửa dữ liệu bị chậm và chiếm dụng bộ nhớ |
| 26 | **Q:** Các dạng tham số ở SP?  🡺 **A:**   |  |  |  | | --- | --- | --- | | IN | OUT | INOUT | | - Tham số tương đương truyền tham trị (mặc định)  - Lấy tham số đầu vào và không bị thay đổi bên ngoài | - Tham số có mục đích lấy giá trị trả về  - Tham số nhận vào giá trị null mặc dù ta truyền có giá trị, nếu bên trong procedure làm thay đổi biến đó thì bên ngoài cũng thay đổi | - Tham số tương đương truyền tham chiếu  - Kết hợp IN và OUT, nhận tham số đầu vào và nếu bên trong procedure làm thay đổi biến đó thì bên ngoài cũng thay đổi | |
| 27 | **Q:** Phân biệt SP, Trigger, Function?  🡺 **A:**   |  |  |  | | --- | --- | --- | | SP | Trigger | Function | | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó  - Khi khai báo SP thì sẽ có 3 loại tham số truyền vào:  + IN: Tham số tương đương truyền tham trị (mặc định)  + OUT: Tham số có mục đích lấy giá trị trả về  +INOUT: Tham số tương đương truyền tham chiếu | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó, được gắn liền trên table nhất định (trigger chỉ áp dụng cho 1 table)  - Trigger sẽ được tự động chạy khi có sự kiện thay đổi dữ liệu trên table (insert, update, delete) | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó nhưng luôn luôn trả về giá trị  - Sử dụng chung được với các câu Query | |
| 28 | **Q:** CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên?  🡺 **A:**  - CSS (Cascading Style Sheets): giúp mô tả cách hiển thị của các thẻ HTML trên các thiết bị  - Có 3 cách để nhúng CSS vào HTML  + Inline: sử dụng thuộc tính style ở HTML  + Internal: sử dụng thẻ style trong thẻ head  + External: sử dụng thẻ link để nhúng 1 file css từ ngoài vào  - Độ ưu tiên: Inline 🡺 Internal & External tùy thuộc vào khai báo, cái nào khai báo sau thì độ ưu tiên cao hơn |
| 29 | **Q:** Phân biệt giữa bộ chọn class và bộ chọn id?  🡺 **A:**   |  |  | | --- | --- | | Class | Id | | - Khai báo một class, các thẻ sẽ có cùng chung thuộc tính trong CSS  - Có dấu **.** ở đầu | - Khi khai báo id cho 1 thẻ nào đó thì tên id đó không được dùng lại nx  - Có dấu # ở đầu | |
| 30 | **Q:** Trình bày mô hình hộp (box model) trong CSS?  🡺 **A:**  - Content: là phần text hoặc hình ảnh ở trên màn hình  - Padding: khoảng trống nằm giữa content và border, bọc xung quanh content  - Border: phần khung bao bọc xung quanh padding và content  - Margin: phần ngoài cùng của Box Model |
| 31 | **Q:** Trình bày thuộc tính float, clear?  🡺 **A:**   |  |  | | --- | --- | | Float | Clear | | - Sử dụng để chuyển một phần tử sang góc trái hoặc phải của không gian bao quanh nó, cần thiết trong việc đinh dạng bố cục trang | - Ngăn chặn thành phần này chiếm vùng không gian của thành phần khác. Đôi khi không muốn float ở một số tình huống nào đó, sẽ sử dụng clear để khắc chế. | |
| 32 | **Q:** Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed?  🡺 **A:**  - Thuộc tính position quy định cách thức xếp đặt vị trí của 1 thành phần trong trang Web  - Phân biệt:   |  |  |  |  | | --- | --- | --- | --- | | Static | Absolute | Relative | Fixed | | - Phần tử sẽ được sắp xếp đúng theo vị trí tự nhiên của nó | - Xác định vị trí tuyệt đối so với thành phần bên ngoài hoặc cửa số trình duyệt | - Xác định vị trí tương đối theo thành phần bên ngoài, bị ảnh hưởng bởi left, top,... | - Giúp thành phần cố định 1 chỗ, không ảnh hưởng khi scrall màn hình | |
| 33 | **Q:** Responsive Web Design là gì?  🡺 **A:**  - Là khái niệm thể hiện trang Web trên tất cả các loại thiết bị với đầy đủ nội dung và không xuất hiện trên thanh scrollbar nằm ngang |
| 34 | **Q:** Làm thế nào để tạo 1 trang web responsive?  🡺 **A:**  - Dùng media queries để chỉnh sửa css cho thiết bị khác nhau  - Dùng grid view: 12 cột |
| 35 | **Q:** Bootstrap là gì?  🡺 **A:**  - **L**à một bộ công cụ mạnh mẽ – một tập hợp các công cụ HTML, CSS và JavaScript để tạo và xây dựng các trang web và ứng dụng web. |
| 36 | **Q:** Một số thành phần thông dụng ở Bootstrap?  🡺 **A:**  - Bao gồm:  + Carousel: dùng hiển thị slide ảnh dạng quảng cáo  + Model: hiển thị hộp thoại thông báo – xác nhận  + Tables: dùng để hiển thị danh sách của đối tượng  + Form: dùng để hiển thị form input nhập dữ liệu vào |
| 37 | **Q:** Phân biệt Static web và Dynamic web?  🡺 **A:**   |  |  | | --- | --- | | Static web | Dynamic web | | - Web tĩnh với nội dung cố định  - Chỉ xác định dựa trên các ngôn ngữ Front-end, HTML + CSS + JS | - Web động với nội dung thay đổi theo tương tác của người dùng  - Các trang web được xây dựng trên 2 dòng ngôn ngữ:  + Front – end: HTML + CSS + JS  + Back – end: JV (PhP, C#, Python, ...) | |
| 38 | **Q:** Phân biệt 2 phương thức GET/POST  🡺 **A:**   |  |  | | --- | --- | | GET | POST | | - Kém bảo mật | - Bảo mật hơn | | - Data sẽ hiển thị trên thanh URL | - Data không hiển thị trên thanh URL | | - Các hành động ở phương thức GET có thể access hiển thị trên URL | - Các hành động ở phương thức POST chỉ có thể submit từ form | | - Độ dài dữ liệu tối đa của GET là 2048 | - Độ dài dữ liệu không hạn chế | | - Data được gửi dưới dạng text | - Data có thể gửi dưới dạng (âm thanh, hình ảnh, ...) | |
| 39 | **Q:** Phân biệt request/response  🡺 **A:**   |  |  | | --- | --- | | Request | Response | | - Là yêu cầu gửi từ Client 🡺 Server | - Là phản hồi từ Server 🡺 Client | |
| 40 | **Q:** Phân biệt forward/redirect  🡺 **A:**   |  |  | | --- | --- | | Forward | Redirect | | - Là chuyển tiếp từ Servelet sang trang JSP kèm data | - Là chuyển hướng từ servelet này sang servelet khác hoặc có thể ra ngoài website nhưng không kèm data | |
| 41 | **Q:** 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 ?  🡺 **A:**  - **Là JSP (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.  - Về bản chất, JSP cũng chính là Servlet. Vì trong quá trình biên dịch, JSP sẽ được chuyển thành Servlet rồi Servlet này sẽ được biên dịch. Servlet của trang JSP có vòng đời giống như các Servlet thông thường.  - Phân biệt JSP và Servlet   |  |  | | --- | --- | | JSP | Servlet | | - Là mã dựa trên HTML | - Là mã Java | | - Viết dễ | - Viết khó hơn | | - Bước đầu tiên, nó sẽ dịch mã Java rồi biên dịch nên làm việc chậm hơn | - Làm việc nhanh hơn | | - Chỉ chấp nhận duy nhất yêu cầu của giao thức HTTP | - Chấp nhận được tất cả yêu cầu của các giao thức | | - Bật tự động quản lý phiên | - Người dùng phải thực hiện thao tác bật quản lý phiên | | - Không cho phép người dùng ghi đè lên phương thức service() | - Cho phép | | - Có khả năng sửa lỗi nhanh | - Sửa lỗi chậm vì phải trải qua các hoạt động như tải lại, tái biên dịch và tái khởi động máy chủ | |
| 42 | **Q:** JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ?  🡺 **A:**  - Là bộ thư viện thẻ được xây dựng dựa trên ngôn ngữ Java dùng trên JSP  - Các thẻ thông dụng:  + <c:out>: in dữ liệu ra màn hình dạng text  + <c:set>: để tạo 1 biến chứa dữ liệu trong jsp  + <c:remove>: xóa biến đã tạo ở <c:set>  + <c:forEach>: lặp để thực hiện hiển thị dữ liệu dạng list  + <c:if>: (rẽ nhánh) hiển thị dựa trên điều kiện trong if  + <c:choose>, <c:when>, <c:otherwise>: sử dụng giống như switch – case |
| 43 | **Q:** Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC?  🡺 **A:**  - Gồm 3 phần chính: Model, View, Controller  + Model: là thành phần thực hiện xử lý nghiệp vụ và tương tác trực tiếp với database  + View: là thành phần giao diện để người dùng tương tác với hệ thống  + Controller: là thành phần điều hướng, nhận, gửi dữ liệu từ View và Model |
| 44 | **Q:** Sự khác nhau giữa MVC1 và MVC2  🡺 **A:**   |  |  | | --- | --- | | MVC1 | MVC2 | | - Controller và View đều là jsp. Yêu cầu từ Client sẽ được gửi tới trang JSP ban đầu được coi là controller xử lý, sau đó truyền lệnh thực thi đến các JavaBeans lấy dữ liệu từ database trả về trang JSP ban đầu theo yêu cầu phía Client  - Là thế hệ đầu tiên sử dụng JSP/JavaBeans để thực hiện mô hình MVC | - Các controller là các Servlet còn Model là các class của java, view là trang JSP  - Là mô hình kiến trúc MVC trên nền web  - Làm việc hiệu quả, logic hơn | |
| 45 | **Q:** Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC?  🡺 **A:**  - B1: Kết nối với database  - B2: Tạo đối tượng statement  - B3: Thực thi câu lệnh SQL:  + Với câu lệnh select thì executeQuery 🡺 trả về đối tượng ResultSet  + Với insert, update, delete thì executeUpdate 🡺 trả về số record bị ảnh hưởng |
| 46 | **Q:** Phân biệt các loại Statement ?  🡺 **A:**   |  |  |  | | --- | --- | --- | | Statement | PrepareStatement | CallableStatement | | - Dùng tạo câu truy vấn tĩnh 🡺 dễ có lỗ hổng bảo mật | - Kế thừa từ Statement, dùng để tạo câu SQL động, ngăn chặn được SQL injection | - Kế thừa từ Prepare Statement và dùng để gọi tới Store Procedure hoặc Function | |
| 47 | **Q:** Kỹ thuật sử dụng transaction bằng JDBC là như thế nào?  🡺 **A:**  - Thiết lập auto\_commit là false. Khi đã thực hiện các thay đổi với cơ sở dữ liệu và muốn ký thác các thay đổi đó, thì sử dụng phương thức commit(). Hoặc muốn xóa các thay đổi đã được thực hiện trước đó để quay về trạng thái trước khi thực hiện thay đổi khi thấy rằng có lỗi xảy ra, sử dụng phương thức rollback(). |
| 48 | **Q:** Các tính chất của Transaction?  🡺 **A:**   |  |  | | --- | --- | | Automicity | - Tất cả đều thành công hoặc tất cả đều thất bại | | Consistency | - Bảo đảm tính đồng nhất dữ liệu | | Isolation | - Bảo đảm Transaction này là độc lập với Transaction khác | | Durability | - Khi một Transaction đã được ký thác thì nó sẽ vẫn tồn tại như thế cho dù xảy ra các lỗi | |
| 49 | **Q:** Phương pháp gọi 1 SP ở JDBC?  🡺 **A:**  - Tạo CallableStatement  + Đối với procedure không có tham số: {call procedure\_name}  + Đối với procedure có tham số: {call procedure\_name(?, ?, ?...)}  + Đối với procedure có giá trị trả về: {?= call procedure\_name(?, ?, ?...)} |
| 50 | **Q:** Phân biệt Ant, Maven, Gradle?  🡺 **A:**   |  |  |  | | --- | --- | --- | | Ant | Maven | Gradle | | - Là công cụ được sử dụng để quản lý thư viện ở Project JSP Servlet. Cách quản lý của Ant có khả năng thủ công và phức tạp | - Ra đời khắc phục các nhược điểm của Ant, nó sử dụng file cấu hình pom.xml để quản lý thư viện. Flie pom.xml này vẫn còn dài dòng do sử dụng xml | - Ra đời khắc phục các nhược điểm của Ant và Maven, nó hỗ trợ được nhiều ngôn ngữ hơn. Gradle sử dụng cấu hình build.gradle để quản lý thư viện. Flie này sử dụng ngôn ngữ Groovy ngắn gọn hơn Maven | |

**CÂU HỎI AUDIT MODULE 4**

|  |  |
| --- | --- |
| **STT** | **Q&A** |
| 1 | **Q:** Vẽ và trình bày cách Spring MVC xử lý request  🡺 **A:**  - Request từ người dùng gửi tới ứng dụng web, DispatcherServlet sẽ là đối tượng nhận request. Điều hướng request tới handler phù hợp, các Controllers sẽ xử lý request. Chuẩn bị model và chọn view hiển thị. Trả về kết quả xử lý request cho DispatcherServlet. DispatcherServlet sẽ gọi View Template phù hợp để xử lý việc hiển thị trên giao diện bằng cách sử dụng model. View template trả kết quả về cho DispatcherServlet. Trả response về cho người dùng. |
| **2** | **Q:** Trình bày cơ chế Dependency Injection trong Spring  🡺 **A:**  - Các container hoặc framework tạo các đối tượng và cung cấp cho những nơi cần chúng |
| 3 | **Q:** Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?  🡺 **A:**  - Constructor-Based: Container sẽ gọi một constructor với các tham số, mỗi tham số đại diện cho một dependency  - Setter-Based: Container gọi các phương thức setter để khởi tạo bean  - Field-Based: Sử dụng annotation @Autowired để tiêm các dependency |
| 4 | **Q:** Framework là gì ? Framework khác Library chỗ nào ?  🡺 **A:**  - Là phần mềm có tính trừu tượng cao, giúp người dùng có thể xác định các ứng dụng 1 cách nhanh chóng, có thể tùy biến để tạo ra các ứng dụng khác nhau  - Đặc điểm khác biệt giữa Framework và Library đó là cơ chế IoC. Framework sẽ là người nắm quyền điều khiển của chương trình, còn đối với Library thì Library sẽ được gọi bởi code của người dùng và chịu sự điều khiển của luồng code đó |
| 5 | **Q:** Spring Framework là gì ?  🡺 **A:**  - Là 1 framework dùng để xác định các ứng dụng Web = ngôn ngữ Java hoặc Kotlin, thông thường được sử dụng cho các dự án lớn |
| 6 | **Q:** Lợi ích của Spring Framework?  🡺 **A:**  - Chỉ cần quan tâm đến những gì cần làm  - Dễ dàng kiểm tra và thử nghiệm.  - Cung cấp API giúp thuận tiện trong việc dịch các ngoại lệ  - Spring framework được xây dựng theo mô hình MVC nên các tinh năng thay thế các web framework khác đều được cung cấp đầy đủ |
| 7 | **Q:** Nguyên lý đảo ngược quyền điều khiển (Inversion of Control) là gì ?  🡺 **A:**  - Là một nguyên lý thiết kế trong công nghệ phần mềm trong đó các thành phần dựa vào nó để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hướng thủ thục truyền thống. |
| 8 | **Q:** Bean là gì?  🡺 **A:**  - Là các đối tượng tạo thành xương sống của ứng dụng và được quản lý bởi Spring IoC container. Một bean là một đối tượng được khởi tạo, lắp ráp, và được quản lý bởi một Spring IoC container. Các bean này được tạo ra bằng siêu dữ liệu cấu hình mà bạn cung cấp cho container |
| 9 | **Q:** Trong Spring có bao nhiêu Bean Scope?  🡺 **A:**  **-** Singleton:Là scope mặc định, chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container.  - Prototype: Một thể hiện của bean sẽ được tạo cho mỗi lần được yêu cầu(request)  - Request: giống với prototype scope, tuy nhiên nó dùng cho ứng dụng web, một thể hiện của bean sẽ được tạo cho mỗi HTTP request.  - Session: Mỗi thể hiện của bean sẽ được tạo cho mỗi HTTP Session  - Global-Session: Được sử dụng để tạo global sesion bean cho các ứng dụng Portlet. |
| 10 | **Q:** @Autowire là gì?  🡺 **A:**  - Là một annotation của Spring, nó đánh dấu rằng biến này sẽ được init bởi bean container của Spring nếu có phần tử tương ứng. |
| **11** | **Q:** @Component có ý nghĩa gì?  🡺 **A:**  - Được sử dụng để định nghĩa một khuôn mẫu chung cho toàn bộ dự án. Class được đánh dấu với annotation @Component khi dự án Spring được chạy nó sẽ quét qua và tạo 1 bean cho class đó. |
| 12 | **Q:** Trình bày ý nghĩa của Controller  🡺 **A:**  - Là một thành phần trong mô hình MVC dùng để điều hướng và xử lý request, gọi model để xử lý nghiệp vụ, kết nối dữ liệu và gửi dữ liệu lại cho view |
| 13 | **Q:** Trình bày ý nghĩa của ModelAndView Interface  🡺 **A:**  - Là sự kết hợp của 2 mục đích ModelMap and View. Nó cho phép controller trả về 1 giá trị bao gồm Model và View. |
| 14 | **Q:** Trình bày ý nghĩa của ModelMap  🡺 **A:**  - Được sử dụng để truyền dữ liệu từ Controller sang View để hiển thị. Có thể sử dụng ModelMap như một tham số trong method của Controller. ModelMap cài đặt lớp Map interface nên nó thêm các phương thức của Map. |
| 15 | **Q:** Trình bày ý nghĩa của ViewResolver Interface  🡺 **A:**  - Controller do người dùng định nghĩa trả về tên của view và các thông tin chi tiết khác tới Front Controller. Tên của view được trả về có kiểu là String, nên cần xử lý tiếp bởi ViewResolver. ViewResolvers có thể sử dụng để trả lại view. |
| 16 | **Q:** Phân biệt POST với PUT thường sử dụng để làm gì?  🡺 **A:**   |  |  | | --- | --- | | POST | PUT | | - Để tạo dữ liệu  - Khi ấn submit bao nhiêu lần đi chăng nữa thì dữ liệu cũng chỉ tính như submit 1 lần.  - Hệ thống tự tạo id cho dữ liệu | - Để cập nhật lại dữ liệu  - Khi submit quá nhiều lần thì có thể nó sẽ cho ra những kết quả khác nhau  - Khi tạo dữ liệu mới sẽ đi kèm theo với id để hệ thống sẽ tạo id của dữ liệu theo id đó dễ gây trùng lặp dẫn đến dữ liệu bị thay đổi | |
| 17 | **Q:** @RequestMapping làm gì?  🡺 **A:**  - Là annotation dùng để quy định cho class và method được sử dụng để ánh xạ đến các request. |
| 18 | **Q:** Thuộc tính consumes trong các Request Mapping là gì ?  🡺 **A:**  - consumes: chỉ chấp nhận các request có content-type giống với giá trị khai báo bên trong consumes. Có thể sử dụng đối tượng với Media Type được yêu cầu bằng cách sử dụng phần tử consumes của @RequestMapping kết hợp với annotation @RequestBody. |
| 19 | **Q:** Thuộc tính produces trong các Request Mapping là gì ?  🡺 **A:**  - Produces: kiểu dữ liệu trả về, thường dùng với các REST-API. Để tạo đối tượng trong Media Type được yêu cầu, sử dụng phần tử produces của @RequestMapping kết hợp với annotation @ResponseBody. |
| 20 | **Q:** Trình bày cơ chế Data Binding trong Spring  🡺 **A:**  - Là cơ chế liên kết dữ liệu đầu vào (đầu ra) với đối tượng model cho phép tương tác với dữ liệu dễ dàng hơn. Các form đều được liên kết với một đối tượng biểu diễn dữ liệu ở phía sau. Khi thay đổi dữ liệu của đối tượng thì dữ liệu trên các điều khiển cũng thay đổi theo.  **\*Chú ý:** Dữ liệu do người dùng nhập vào thông qua DataBinder trả về kết quả là BindingResult. DataBinder bao gồm các bộ xử lý PropertyEditors, bộ định dạng Fomatters, bộ lọc dữ liệu Validators và bộ chuyển đổi ConversionService |
| 21 | **Q:** So sánh RequestParam và PathVariable ?  🡺 **A:**   |  |  | | --- | --- | | RequestParam | PathVariable | | - Có thể lấy dữ liệu từ form  - Lấy dữ liệu theo kiểu key - value | - Chỉ có thể lấy dữ liệu trên URL  - Chỉ lấy dữ liệu theo value | |
| 22 | **Q:** Thymeleaf là gì?  🡺 **A:**  - Là bộ công cụ dùng để xử lý view(template engine) dựa trên ngôn ngữ Java |
| 23 | **Q:** Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thể nào?  🡺 **A:**  - Cú pháp sử dụng vòng lặp trong Thymeleaf:  <**someHtmlTag** th:each="item : ${items}">  ....  </**someHtmlTag**>  - Cú pháp sử dụng điều kiện trong Thymaleaf:  <**someHtmlTag** th:if="condition">  ......  </**someHtmlTag**> |
| 24 | **Q:** Formatter là gì ? Converter là gì ?  🡺 **A:**  - Formatter là một đối tượng được dùng để chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác, chỉ được thiết kế sử dụng ở tầng web  - Converter là một đối tượng được dùng để chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác, được sử dụng cho toàn bộ hệ thống, sử dụng ở bất kì tầng nào của ứng dụng |
| 25 | **Q:** ORM là gì ?  🡺 **A:**  - (Object Relational Mapping) là kỹ thuật liên kết giữa các đối tượng trong lập trình với đối tượng trong CSDL |
| 26 | **Q:** JPA là gì? Spring Data JPA là gì ?  🡺 **A:**  - JPA (Java Presistance API) là tập hợp các interface dùng cho lập trình ứng dụng Java  - Spring Data JPA là framework giúp cho việc sử dụng JPA trong ứng dụng Spring dễ dàng hơn. Cung cấp các chức năng cơ bản như: CRUD, phân trang, search, sort |
| 27 | **Q:** Hibernate là gì ? Hibernate làm gì với database  🡺 **A:**  - Là ORM Framework triển khai những gì mà JPA đã thiết kế  - Hibernate cho phép truy vẫn dữ liệu thông qua Java Persistence API (JPA) hoặc bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần (Native SQL). |
| 28 | **Q:** Liệt kê một số annotation của hibernate?  🡺 **A:**  - @Entity được sử dụng để chú thích một class là một Entity.  - @Table cho phép chú thích tên bảng thông qua thuộc tính name  - @Column được sử dụng để chỉ định thông tin chi tiết của cột  - @Id được sử dụng để mô tả đây là Id (Identity) của Entity, nó tương đương với cột đó là khóa chính (Primary Key) của table trong database.  - @GeneratedValue được sử dụng để Hibernate tự động tạo ra giá trị và gán vào cho một cột  - @ManyToOne mô tả một quan hệ  **N-1**  **-** @OneToMany mô tả quan hệ **1-N**  **-** @OneToOne mô tả quan hệ **1-1**  **-** @ManyToMany mô tả quan hệ **N-N** |
| 29 | **Q:** Entity là gì?  🡺 **A:**  - Là đối tượng đại diện cho dữ liệu trong ứng dụng. Một entity thông thường sẽ là 1 POJO (Plain Old Java Object) |
| 30 | **Q:** Persistence Context & Entity Manager là gì?  🡺 **A:**   |  |  | | --- | --- | | Persistence Context | Entity Manager | | - Là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu | - Là một interface cung cấp các API cho việc tương tác với các Entity. | |
| 31 | **Q:** Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?  🡺 **A:**  - Câu lệnh truy vấn động hay còn gọi là HQL. EntityManager cung cấp phương thức createQuery() để tạo các câu lệnh truy vấn động  - Câu lệnh truy vấn tĩnh hay còn gọi là Native SQL (SQL thuần). Sử dụng phương thức createNamedQuery của EntityManager để tạo câu truy vấn tĩnh. |
| 32 | **Q:** Trạng thái của Entity bao gồm những gì?  🡺 **A:**  - Added: Thực thể được đánh dấu là đã thêm  - Deleted: Thực thể được đánh dấu là đã xóa  - Modified: Thực thể đã được sửa đổi  - Unchanged: Thực thể chưa được sửa đổi  - Detached: Thực thể không được theo dõi |
| 33 | **Q:** Có bao nhiêu loại mapping trong Hibernate?  🡺 **A:**  - Collections Mappings, Association Mappings, Component Mappings |
| 34 | **Q:** Để sử dụng interface JpaRepository cần cung cấp những thông tin nào?  🡺 **A:**  - Cung cấp đối tượng Entity và kiểu dữ liệu trường khóa chính của đối tượng |
| 35 | **Q:** Validation dữ liệu là gì?  🡺 **A:**  - Là quá trình xác minh dữ liệu từ người dùng, đảm bảo dữ liệu là đúng, là hợp lệ |
| 36 | **Q:** Trình bày cách triển khai validate dữ liệu trong Spring  🡺 **A:**  - Có 3 cách:  + Sử dụng annotation được cung cấp sẵn  + Custom validate  + Tự tạo ra các annotation |
| 37 | **Q:** Binding Result là gì?  🡺 **A:**  - Là một giao diện chỉ định cách đối tượng lưu trữ, kết quả xác thực sẽ lưu trữ và truy xuất kết quả xác thực (lỗi, cố gắng liên kết với các trường không được phép, ...) |
| 38 | **Q:** AOP là gì? AOP gồm những yếu tố nào?  🡺 **A:**  - AOP (Aspect Oriented Programming) lập trình hướng khía cạnh, là phương pháp lâp trình nhằm phân tách chương trình thành các module riêng lẻ, không phụ thuộc vào nhau  - Trong AOP sẽ chia chương trình thành 2 loại concern khác nhau:  + Core\_concern / Primary Concern: đây là requirement là nghiệp vụ chính của chương trình  + Cross\_cutting\_corcern: là những logic xử lý phụ cần được thực hiện của chương trình khi core\_concern được gọi như: bảo mật, ghi log, theo dõi server,... |
| 39 | **Q:** Các loại Advice?  🡺 **A:**  - Có 5 loại:  + Before Advice:  + After Advice:  + Around Advice: thêm cả trước và sau  + After Returning Advice: thêm ngay khi phương thức được return  + After Throwing Advice: thêm ngay khi ngoại lệ được ném ra |
| 40 | **Q:** JoinPoint là gì?  🡺 **A:**  - Là một điểm trong chương trình, là những nơi có thể chèn cross-cutting-concern |
| 41 | **Q:** Nêu cách triển khai AOP mà bạn dùng trong dự án  🡺 **A:**  - Khởi chạy Spring Initializr, chọn Spring Web, Thymeleaf, Spring Data JPA, MySQL Driver, Validation |
| 42 | **Q:** Mục đích xử lý ngoại lệ trong Spring Web MVC ?  🡺 **A:**  - Khi một ngoại lệ xảy ra chương trình sẽ bị ngắt. Khi đó, tất cả các nguồn tài nguyên mà hệ thống đã cấp không được giải phóng. Điều này gây lãng phí tài nguyên. |
| 43 | **Q:** Các cách xử lý ngoại lệ trong Spring Web MVC ?  🡺 **A:**  **-** Sử dụng try-catch để xử lý các ngoại lệ tại request  - Sử dụng @ExceptionHandler để ủy quyền cho front controller xử lý ngoại lệ.  - Sử dụng HandlerExceptionResolver để xử lý ngoại lệ tại servlet.  - Sử dụng trang báo lỗi để bắt các fatal error hay các ngoại lệ sinh ra ngoài phạm vi của Spring. |
| 44 | **Q:** Cookie là gì?  🡺 **A:**  - Là tệp được tạo ra khi người dùng truy cập vào các website. Cookie được lưu trữ vào Client nhằm tăng tính trải nghiệm người dùng, ghi nhớ thông tin đăng nhập, lưu thông tin người dùng |
| 45 | **Q:** Phân biệt Session và Cookie  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | Session | Cookie | | Giống nhau | - Đều tạo ra dữ liệu trong quá trình người dùng sử dụng | | | Khác nhau | - Được lưu ở server  - Thời gian sống phụ thuộc vào phiên làm việc  - Có thể lưu trữ các kiểu dữ liệu của server  - Hoạt động dựa vào Cookie  - Dung lượng vô hạn | - Được lưu ở Client  - Thời gian sống phụ thuộc vào sự cài đặt của LTV  - Chỉ lưu trữ được dữ liệu text  - Dung lượng tối đa 4KB | |
| 46 | **Q:** Session là gì ? Cho ví dụ về session ?  🡺 **A:**  - Là phiên làm việc giữa client và server. Một session bắt đầu khi client thực hiện request đầu tiên lên server và kết thúc khi client dừng làm việc với server  - Ví dụ: Thông thường sử dụng session để lưu thông tin đăng nhập, giỏ hàng hoặc những dữ liệu mang tinh tạm thời |
| 47 | **Q:** Nêu cách triển khai Session mà bạn dùng trong dự án  🡺 **A:**  - Phương pháp 1: Inject trong HttpSession nơi được yêu cầu – Sử dụng annotation @Autowired để tiêm phụ thuộc đối tượng httpSession. Ví dụ: Để hiện thị student trong httpsession, chúng ta có GetMapping với Request là viewSession, lấy student từ httpsession ra sẽ có Student student = httpSession.getAttibute(attribute tên là student), lúc đó sẽ lấy ra được một object là student và cần phải ép kiểu, thêm student vào ModelAndView  - Phương pháp 2: Sử dụng như một tham số - Thêm HttpSession vào trong @RequestMapping  - Phương pháp 3: Tạo một bean và giới hạn cho session làm việc  - Phương pháp 4: Sử dụng annotation @SessionAttributes |
| 48 | **Q:** Vì sao Session hoạt động dựa vào Cookie  🡺 **A:**  - Khi truy cập trang web, Server sẽ tạo ngẫu nhiên 1 ID định danh cho phiên hoạt động và đồng thời máy cũng tự sinh 1 cookie lưu giá trị của Session ID đó. Viêc này giúp Server quản lý được các request gửi lên để có thể trả respone về cho đúng client đã request. |
| 49 | **Q:** Web Service là gì? Lấy ví dụ về web service  🡺 **A:**  - Là các thành phần ứng dụng được hiển thị dưới dạng dịch vụ, có thể được tích hợp với các ứng dụng viết bằng cách ngôn ngữ khác nhau  - Ví dụ: hệ thống xử lý đơn đặt hàng và quản lý tài khoản. Nhân viên sử dụng ứng dụng khách hàng để tạo tài khoản mới và nhập các đơn đặt hàng của khách hàng  mới. Logic xử lý cho hệ thống này được viết bằng CSDL để lưu trữ thông tin. |
| 50 | **Q:** Phân biệt Web Service và Web truyền thống (Website)  🡺 **A:**   |  |  | | --- | --- | | Website | Web Service | | - Có giao diện  - Con người sử dụng  - Sử dụng được đa nền tảng  - Tương tác bằng button, input, form...  - Ứng dụng đầu cuối | - Không có giao diện  - Được sử dụng bởi các ứng dụng khác  - Độc lập về nền tảng  - Thông qua các giao thức HTTP: GET, POST, PUT, DELETE  - Được sử dụng để làm back-end, xử lý nghiệp vụ và các phần chức năng cho front-end | |
| 51 | **Q:** SOAP là gì? RESTful là gì?  🡺 **A:**  - SOAP(Simple Object Access Protocol) là giao thức sử dụng XML để định nghĩa và giao tiếp dữ liệu thông qua HTTP  - RESTful là các webservice được xây dựng dựa trên REST |
| 52 | **Q:** jQuery là gì? Ajax là gì?  🡺 **A:**  - jQuery là thư viện của JS, slogan “write less, do more”, giúp cho việc phát triển ứng dụng bằng JS nhanh hơn và code ngắn gọn hơn  - AJAX (Asynchrous Javascript and XML) là một nhóm các công nghệ cho LTV phát triển các ứng dụng web có thể cập nhập được dữ liệu mà không cần load lại trang |
| 53 | **Q:** Nêu ví dụ về cách bạn triển khai Ajax trong dự án của bạn  🡺 **A:**  - Vào build.gradle để thêm dependences jackson-core và jackson-databind và cấu hình thư viện sử dụng ajax vào file HTML |
| 54 | **Q:** @RequestHeader và @ResponseHeader có ý nghĩa gì?  🡺 **A:**  - @RequestHeader là các thông số kèm theo khi gửi yêu cầu lên server  - @ResponseHeader là các thông số kèm theo khi server gửi kết quả về cho client. |
| 55 | **Q:** I18n và L10n là gì?  🡺 **A:** |
| 56 | **Q:** Nêu cách triển khai I18N và I10N trong dự án Spring  🡺 **A:** |
| 57 | **Q:** Interceptor là gì?  🡺 **A:** |
| 58 | **Q:** Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?  🡺 **A:** |
| 59 | **Q:** Authentication là gì ? Authorization là gì ?  🡺 **A:**  - Authentication hay còn gọi là xác thực là quá trình chứng thực thông tin của người dùng là đáng tin cậy. Xác thực cũng có nghĩa là chúng ta công nhận nguồn gốc của đối tượng  - Authorization hay còn gọi là phân quyền, là quá trình xác định xem một người dùng nào đó có được sử dụng những tính năng và quyền truy cập mà ứng dụng cho phép. Việc cấp quyền không thể được sử dụng mà không có sự xác thực |
| 60 | **Q:** Liệt kê các cơ chế xác thực  🡺 **A:**  - HTTP Basic, Cookies, One Time Password, JWT: Json Web Token, Signatures: chữ kí số, OAuth2: thông qua bên thứ 3 |
| 61 | **Q:** CSRF là gì  🡺 **A:** |
| 62 | **Q:** CORS là gì  🡺 **A:** |
| 63 | **Q:** Spring Boot là gì?  🡺 **A:**  - Là một trong số các module của [Spring framework](https://james.codegym.vn/mod/page/view.php?id=1746), cung cấp tính năng RAD (Rapid Application Development), được sử dụng để tạo ra một ứng dụng độc lập dựa trên Spring có thể chạy ngay với rất ít cấu hình cần thêm. |
| 64 | **Q:** Sự khác nhau giữa SpringBoot và SpringMVC là gì?  🡺 **A:**   |  |  | | --- | --- | | Spring Boot | Spring MVC | | - Là một module của Spring để đóng gói ứng dụng dựa trên Spring với các giá trị mặc định  - Cung cấp các cấu hình mặc định  - Không có yêu cầu đối với các file mô tả khi triển khai  - Tránh mã viết sẵn và kết hợp các dependencies lại với nhau trong một đơn vị duy nhất  - Giảm thời gian phát triển và tăng năng suất | - Là based web framework dựa trên Spring  - Cung cấp các tính năng có sẵn  - Cần có nhiều file mô tả triển khai  - Chỉ định từng dependency riêng biệt  - Cần nhiều thời gian để đạt được năng suất cao | |
| 65 | **Q:** Cấu hình cho Spring Boot Tìm kiếm các Bean ở nhiều package khác nhau bằng cách nào?  🡺 **A:** |

**CÂU HỎI AUDIT MODULE 5**

|  |  |
| --- | --- |
| **STT** | **Q&A** |
| 1 | **Q:** TypeScript là gì?  🡺 **A:**  - Là một NNLT được phát triển bởi Microsoft  - Là superset của JS  - LTV có thể transpile từ TS sang JS.  - TS ra đời nhằm hỗ trợ thêm cho JS các phần như OOP, DataType, Generic,… |
| 2 | **Q:** Tại sao nên sử dụng Typescript, Ưu điểm của Typescript  🡺 **A:**  - Vì nó được thiết kế thêm nhiều chức năng tiện lợi, cải tiến những điểm yếu của JS, có thể hoạt động rộng rãi cho các ứng dụng của ngôn ngữ Angular2 và Nodejs  - Ưu điểm:  + Hỗ trợ quá trình tìm kiếm giúp tiết kiệm thời gian kiểm tra lại code  + Hoàn toàn miễn phí  + Thao tác khá đơn giản, tiết kiệm thời gian, khắc phục tình trạng xuất hiện lỗi, dễ đọc hơn và chỉnh sửa code dễ dàng hơn |
| 3 | **Q:** Làm sao để cài đặt TypeScript  🡺 **A:**  - B1: Cài đặt NodeJS  - B2: Mở cmd  - B3: Gõ lệnh npm -v để in ra version của npm  - B4: Cài đặt TS bằng cách gõ lệnh npm install -g typescript |
| 4 | **Q:** Có thể chuyển mã TypeScript thành mã Javascript không, bằng cách nào?  🡺 **A:**  - Được. Cách chuyển: tsc dương\_dan/ten\_file.ts |
| 5 | **Q:** Kể tên các kiểu dữ liệu trong Typescript  🡺 **A:**  - TS hỗ trợ tất cả các kiểu dữ liệu của JS và bổ sung thêm một số kiểu dữ liệu khác  như: any (một kiểu dữ liệu mà chúng ta không biết chắc chắn kiểu dữ liệu của nó),  interface, enum, tuple (khai báo mảng) |
| 6 | **Q:** Kể tên các vòng lặp được typescript hỗ trợ  🡺 **A:**  - While, for, for … of, for … in, do … while |
| 7 | **Q:** Phân biệt sự khác nhau giữa var, let, const  🡺 **A:**   |  |  |  | | --- | --- | --- | | Var | Let | Const | | - Khai báo những giá trị có thể thay đổi  - Cho phép khai báo lại biến với cùng 1 tên  - Việc khai báo được đẩy lên đầu chương trinh  - Nếu var được khai báo ở block scope, thì vòng đời nằm ở trong lẫn ngoài block scope | - Khai báo những giá trị có thể thay đổi  - Không cho phép khai báo lại biến với cùng một tên  - Không có cơ chế hoisting  - Phạm vi của let chỉ ở block scope | - Dùng để khai báo hằng số, không thay đổi giá trị tham chiếu | |
| 8 | **Q:** Trình bày khái niệm Single page application  🡺 **A:**  - Là một ứng dụng web hay một website mà ở đó tất cả các thao tác của người dùng chỉ diễn ra trên 1 trang duy nhất, tất cả các cấu trúc của trang web (HTML) được load sẵn 1 lần và sẽ không load lại ngay cả khi chuyển trang. |
| 9 | **Q:** Lazy-loading là gì?  🡺 **A:**  - Là kỹ thuật thực hiện trì hoãn tải các tài nguyên không quan trọng vào thời điểm tải trang. Thay vì tải ngay lập tức, các tài nguyên không quan trọng này chỉ tải vào thời điểm cần thiết. |
| 10 | **Q:** Để khởi tạo một dự án angular sử dụng câu lệnh nào ?  🡺 **A:**  - Lệnh: ng new ten\_du\_an |
| 11 | **Q:** Để chạy một dự án angular sử dụng câu lệnh nào ?  🡺 **A:**  - Lệnh: ng s --o |
| 12 | **Q:** Tại sao nên sử dụng Angular (điểm mạnh)?  🡺 **A:**  - Angular giúp nâng cao năng suất của các LTV  - Cấu trúc phát triển rõ ràng  - Hỗ trợ đầy đủ tính năng điều hướng (**routing**)  - Giảm tối đa kích thước và tăng tối đa hiệu suất của ứng dụng. |
| 13 | **Q:** Nhược điểm của Angular là gì?  🡺 **A:**  - Không an toàn. Angular là một trong những Framework Front-end, mà front-end không bảo mật bằng Back-end  - Với một số trình duyệt sở hữu tính năng Disable JavaScript thì website sẽ không thể sử dụng được trên trinh duyệt đó |
| 14 | **Q:** Angular Cli là gì? Làm sao để sử dụng nó ?  🡺 **A:**  - Là một công cụ được phát triển đề chạy trên giao diện cửa sổ ứng dụng nhằm giúp đỡ việc tạo dự án, quản lý tập tin trong dự án và thực hiện nhiều tác vụ khác nhau như test (kiểm thử), bundle và deploy dự án.  - Sử dụng Angular Cli  + B1: Cài đặt môi trường Node.js  + B2: Mở CMD và gõ câu lệnh npm -v  + B3: Chạy câu lệnh npm install -g @angular/cli |
| 15 | **Q:** File angular.json và package.json sử dụng để làm gì?  🡺 **A:**  - Package.json chứa các thư viện cần thiết cho dự án Angular, Angular.json dùng để cấu hình Angular CLI |
| 16 | **Q:** Giải thích kiến trúc của Angular  🡺 **A:**  - Component là một class JS thuần và định nghĩa bởi @Component decorator. Decorator này cung cấp cho component thông tin về View (template) và thông tin metadata về class  - Component sử dụng data binding để lấy dữ liệu từ Component sang View (template), sử dụng HTML đặc biệt là Template  - Service cung cấp các service cho Component  - Trách nhiệm cung cấp các thể hiện của Service cho Component gọi là Injector. Nó jnject các service vào component sử dụng Dependency Injection  - Sử dụng Module để tổ chức code ứng dụng Angular. Ứng dụng sẽ gộp lại bởi nhiều Module. Mỗi Module triển khai một tinh năng cụ thể hoặc các tinh năng của ứng dụng. |
| 17 | **Q:** Angular đi theo mô hình nào? Phân biệt mô hình MVVM và MVC  🡺 **A:**  - Angular đi theo mô hình MVVM  - Phân biệt:   |  |  | | --- | --- | | MVC | MVVM | | - Là một mô hình kiến trúc phân tách ứng dụng thành 3 phần riêng biệt Model, View, Controller.  - Triển khai ở máy chủ | - Là mô hình Model View View Model  - Triển khai ở phía Client | |
| 18 | **Q:** Component là gì ? Các thành phần của component  🡺 **A:**  - Là thuật ngữ mang ý nghĩa như là bộ phận, thành phần hoặc yếu tố cấu thành của một hệ thống hoặc phần mềm nào đó, được gọi ra từ các chương trình khác, được liên kết và sử dụng như một bộ phận của chương trình.  - Các thành phần của component: metadata, template, class |
| 19 | **Q:** Mô tả vòng đời component?  🡺 **A:**  - Khi ứng dụng Angular được start lên thì đầu tiên nó sẽ tạo và render component cha (hay còn gọi là root component) sau đó nó sẽ tạo và render các component con. Khi mỗi component được load lên, component sẽ kiểm tra xem có data binding vào nó không, dữ liệu có thay đổi không và cập nhập lại chúng. Khi component bị phá huỷ thì chúng sẽ bị remove (xoá) khỏi giao diện web.  - Thứ tự từ trên xuống dưới về các method sẽ được sử dụng trong vòng đời của một component.  + ngOnChanges  + ngOnInit  + ngDoCheck  + ngAfterContentInit  + ngAfterContentChecked  + ngAfterViewInit  + ngAfterViewChecked  + ngOnDestroy |
| 20 | **Q:** Trình bày được khái niệm template trong angular  🡺 **A:**  **-** Template được xem là bộ khung xương trong website,  chúng được thiết kế và sắp xếp thành một bố cục cụ thể giúp người sử dụng có thể dễ dàng thao tác thay đổi mà không cần biết code. |
| **21** | **Q:** Biên dịch AOT là gì? Những lợi thế của AOT ?  🡺 **A:**  - Biên dịch trước thời gian (AOT) là khi mã được biên dịch như một phần của quá trình xây dựng  - Những lợi thế:  + Với AOT, mã được biên dịch trong quá trình xây dựng nên kết xuất ứng dụng Angular nhanh hơn  + Không cần tải xuống trình biên dịch Angular nên kích thước ứng dụng Angular nhỏ hơn  + Chất lượng mã tốt hơn  + Các ứng dụng an toàn và mạnh mẽ hơn |
| 22 | **Q:** Trình bày về Data binding trong component  🡺 **A:**  - Là kỹ thuật, nơi dữ liệu được đồng bộ giữa component và tầng view (template file html). Data binding trong Angular có thể chia ra làm 2 nhóm. Đó là one way binding (binding 1 chiều) và two way binding (binding 2 chiều). |
| 23 | **Q:** Hãy mô tả khái niệm Databinding One Way và Two Way phân biệt sự khác nhau?  🡺 **A:**   |  |  | | --- | --- | | Data binding One Way | Data binding Two Way | | - Dữ liệu được truyền 1 chiều. Có thể từ view sang component hoặc ngược lại từ component sang view.  - Từ component sang view chúng ta sử dụng Interpolation & Property Binding để hiển thị dữ liệu  - Sử dụng để hiển thị giá trị từ component sang view. | - Là thay đổi dữ liệu từ component qua view và ngược lại từ view chúng ta thay đổi dữ liệu.  - 2 way binding thì hữu dụng khi làm form.  - Sử dụng ngModel để thực hiện việc binding 2 chiều. | |
| 24 | **Q:** Directive trong angular là gì ?  🡺 **A:**  - Là một thành phần của Angular Framework dùng để tăng cường sức mạnh và hỗ trợ thêm các tính năng cho HTML. Directive giúp LTV có thể thay đổi được cấu trúc và thuộc tinh của HTML |
| 25 | **Q:** Kể tên các directive được hỗ trợ trong angular  🡺 **A:**  - Component Directive: bản thân mỗi component chính là một directive  - Structural Directive: là các directive dùng để thay đổi cấu trúc của HTML: “ngFor”, “ngIf”  - Attribute Directive: là các directive dùng để thay đổi thuộc tính của HTML: [style], [class]  - Custom Directive: là directive do LTV tự định nghĩa trong dự án |
| 26 | **Q:** Làm thế nào để lấy được dữ liệu từ component cha  🡺 **A:**  - B1: Đặt @Input cho biến x cần nhận dữ liệu từ component cha  - B2: Sử dụng directive [x] = “value” trong selector của component con |
| 27 | **Q:** Làm thế nào để lấy được dữ liệu từ component con  🡺 **A:**  - C1: Sử dụng @Output và Emitter  + B1: Khai báo @Output cho biến x, có kiểu dữ liệu là EventEmitter  + B2: Gọi hàm emit của Emitter để gửi dữ liệu lên component cha  + B3: Sử dụng (x) = “nameFunction($event)” với event chính là dữ liệu component con gửi lên  - C2: Sử dụng biến cục bộ  - C3: Sử dụng @ViewChild |
| 28 | **Q:** Pipe là gì? Kể tên 1 số pipe thường sử dụng  🡺 **A:**  - Là thành phần của Angular dùng để thay đổi cách hiển thị của dữ liệu trên giao diện (lưu ý: pipe không làm thay đổi dữ liệu gốc)  - Các loại pipe thường dùng: titlecase, datapipe, currency pipe, decimal pipe, … |
| 29 | **Q:** ngFor là gì? Cách sử dụng  🡺 **A:**  - Sử dụng ngFor để duyệt qua các tập hợp dữ liệu như array, list. Sau đó tạo các thành phần web tương ứng với mỗi vòng lặp.  - Cách sử dụng:  <html-element ngFor="let <item> of <items>;">  <html-Template></html-Template>  </html-element>  - Trong đó:  + html-element: là thẻ web muốn sử dụng ngFor directive  + \*ngFor: dấu \* tượng trưng cho cú pháp của Angular  + let of <item>: là giá trị hiện tại của phần tử trong mảng, items chinh là mảng các phần tử |
| 30 | **Q:** ngIf là gì? Cách sử dụng  🡺 **A:**  - Sử dụng ngIf để xóa hoặc thêm một phần tử trên web dựa vào điều kiện có thoả mãn hay không.  - Cách sử dụng:  <p \*ngIf="condition">  các thẻ html sẽ được hiển thị nếu điều kiện là đúng  </p> |
| 31 | **Q:** Các cách tạo form trong Angular  🡺 **A:**  - Có 2 cách để tạo form: Template dirven form và Reactive form |
| 32 | **Q:** Phân biệt Template driven form và Reactive form  🡺 **A:**   |  |  | | --- | --- | | Template driven form | Reactive form | | - import FormModule  - Tạo form từ HTML  - Xử lý theo hướng bất đồng bộ  - Không được khuyến cáo sử dụng | - import ReactiveForm Module  - Tạo form từ TS - Xử lý theo hướng đồng bộ  - Được khuyến cáo nên sử dụng | |
| 33 | **Q:** Trình bày 1 số validate thường dùng. Cách custom validate như thế nào?  🡺 **A:**  - 1 số validate thường dùng:  + required dùng để check trống trường input  + email dùng để yêu cầu dữ liệu điền vào phải là email  + minlength: độ dài nhỏ nhất  + maxlength: độ dài lớn nhất  - Cách custom validate: Import 2 thư viện AbstractControl và ValidationErrors phải có 1 tham số có kiểu là AbstractControl và kiểu trả về của nó là null hoặc ValidationErrors. Với ValidationErrors chứa cặp key và value {key: value} |
| 34 | **Q:** Router trong angular dùng để làm gì?  🡺 **A:**  - Dùng để xác định component nào sẽ được hiển thị |
| 35 | **Q:** Cách để lấy tham số trên URL trong angular  🡺 **A:**  - Sử dụng ActivateRoute và ParamMap để lấy URL |
| 36 | **Q:** Làm sao để kiểm tra người dùng có quyền truy cập vào 1 route ?  🡺 **A:** |
| 37 | **Q:**Trình bày sự khác nhau giữa navigateByUrl và navigate  🡺 **A:**   |  |  | | --- | --- | | navigateByUrl | Navigate | | - Là đường truyền tĩnh | - Là đường dẫn động | |
| 38 | **Q:** Sự khác biệt giữa RouterModule.forChild và RouterModule.forRoot là gì?  🡺 **A:** |
| 39 | **Q:** Trình bày được ý nghĩa và khái niệm Observable  🡺 **A:**  - Là khái niệm đại diện cho một giá trị hoặc sự kiện sẽ phát sinh trong tương lai. Khi các giá trị hoặc sự kiện trong tương lai được trả về, nó sẽ được điều hướng vào observable (observer là một callback function có 3 tham số) |
| 40 | **Q:** Trình bày được ý nghĩa và khái niệm Observer  🡺 **A:**  - Là một behavioral design pattern (mô hình thiết kế hành vi) dùng để định nghĩa các sự kiện xảy ra với đối tượng mà chúng đang quan sát (observe). |
| 41 | **Q:** Phân biệt sự khác nhau giữa Promise và Observable  🡺 **A:**   |  |  | | --- | --- | | Promise | Observable | | - Không có các khả năng như Observable | - Có khả năng trả về nhiều kết quả  - Có khả năng cancel ((hủy request đã gửi đi)  - Có khả năng retry request (gửi lại request)  - Có khả năng xử lý kết quả trước khi trả về (thông qua các hàm map, filler ..) | |
| 42 | **Q:** Trình bày được ý nghĩa và khái niệm subcribe  🡺 **A:**  - Khi làm việc với **Observable** thì sẽ thường xuyên sử dụng phương thức subscribe  để theo dõi các thay đổi của dữ liệu hoặc nhận về kết quả của một công việc nào đó. Khi API trả về kết quả thì dữ liệu đó sẽ được log ra qua phương thức **subscribe.** |
| 43 | **Q:** ngModule là gì?  🡺 **A:**  - **L**à cấu trúc cơ bản khi làm việc với bất kỳ ứng dụng bằng Angular, nhưng nó cũng tương đối tinh vi và phức tạp, vì những scope của nó. Mục đích của NgModule là khai báo từng thứ được tạo trong Angular và nhóm chúng lại với nhau (như các package Java hoặc namespace PHP / C#).  - Là 1 thư viện, người dùng có thể import vào module khác để sử dụng hoặc có thể export cho module khác |