1. Phân biệt các câu lệnh DML & DDL? DML, DDL bao gồm các câu lệnh nào?

* DDL (data definition language) : chứa các câu lệnh định nghĩa cấu trúc dữ liệu. Sử dụng để tạo , sửa xóa các database hoặc table trong database. Bao gồm :
  + Create : tạo mới
  + Alter : sửa
  + Drop : xóa
  + Rename : đổi tên
  + Truncate : xóa hết dữ liệu trong bảng .( không xóa cấu trúc chỉ xóa dữ liệu trong bảng )
* DML (data manipulation language) là các câu lệnh thao tác với dữ liệu trong table của database. Bao gồm :
  + Select : lấy dữ liệu từ bảng
  + Insert : thêm dữ liệu vào bảng
  + Update : cập nhật lại dòng dữ liệu trong bảng
  + Delete : xóa dòng dữ liệu ( hoặc xóa tất cả dữ liệu trong bảng tùy vào điều kiện đưa ra.

2. Cơ sở dữ liệu (database) là gì?

* Là phần nằm cuối cùng ở back-end chứa dữ liệu của 1 trang web hay 1 ứng dụng nào đó.

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 ( entity relationship diagram) là sơ đồ quan hệ giữa các thực thể. ERD sinh ra với mục đích là chuyển hóa những yêu cầu của khách hàng dưới dạng lời nói hoặc giấy tờ thành 1 sơ đồ từ đó thuận tiện cho việc thiết kế database.
* ERD bao gồm :
  + Entity : các thực thể tham gia vào sơ đồ được biểu diễn dưới dạng các hình chữ nhật.
  + Các thuộc tính của entity được biểu diễn dưới dạng các hình elip.
  + Mối quan hệ giữa các thực thể biểu diễn dưới dạng hình thoi.

4. Các bước chuyển đổi từ ERD -> Table?

* B1 : Biến đổi các entity thành table :
  + Các thuộc tính khóa của entity sẽ là khóa chính (primary key ) của table. Thuộc tính khóa của entity sẽ là thuộc tính duy nhất không bị trùng lặp và không thể null.
  + Các thuộc tính thường của entity sẽ là thuộc tính thường của table.
* B2 : Biến đổi các thuộc tính đa trị:
  + Tách các thuộc tính đa trị thành 1 table riêng. Và đặt khóa ngoại vào table hiện tại
* B3 : biến đổi mối quan hệ 1 : 1 :
  + Đặt thuộc tính khóa ở phía tùy chọn
* B4 : biển đối mối quan hệ 1 : nhiều ( 1 : n ):
  + Đặt thuộc tính khóa ở 1 vào nhiều. Ví dụ 1 lớp có nhiều học sinh, mỗi học sinh chỉ học 1 lớp. Đặt khóa của lớp vào học sinh
* B5 : biến đổi mối quan hệ nhiều : nhiều ( n : n ):
  + Sinh ra 1 table mới có khóa chính là khóa chính là tập hợp 2 khóa chinh của 2 thực thể tham gia vào mối quan hệ. Ví dụ 1 lớp dạy bởi nhiều giáo viên và 1 giao viên có thể dạy nhiều lớp . sẽ sinh ra 1 bảng mới có khóa chính là khóa tích hợp 2 khóa chính của thực thể lớp và giáo viên.

5. Các bước chuẩn hóa dữ liệu?

* 1NF ( chuẩn 1 ) : mỗi ô trong mỗi dòng chỉ chứa giá trị duy nhất , giá trị trong cột phải cùng loại , mỗi dòng phải là duy nhất không chứa giá trị lặp. Không chứa các cột có giá trị có thể tính toán từ thuộc tính khác.
* 2NF :
  + Đạt được chuẩn 1 .
  + Tất cả các thuộc tính thường đều phải được suy ra từ khóa chính.
* 3NF :
  + Đạt được chuẩn 2.
  + Không có dữ liệu nào phụ thuộc hàm vào các trường không phải khóa chính

6. Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL?

* Ràng buộc là những quy tắc được áp dụng trên cột dữ liệu của bảng, được sử dụng để kiểm tra tình hợp lệ của dữ liệu vào, đảm bảo tính chính xác , toàn vẹn của dữ liệu
* Bao gồm :
  + Not null : không cho phép dữ liệu có gia trị là null.
  + Default : dùng để gán giá trị mặc định cho một trường dữ liệu.
  + Unique : sử dụng để bảo đảm dữ liệu của cột là duy nhất và không bị trùng lặp giá trị trên cùng 1 cột.
  + Primary key : dùng để xác định khóa chính của bảng. Kết hợp giữa not null và unique
  + Foreign key : dùng để thiết lập khóa ngoại trên bảng tham chiếu đến bảng khác thông qua giá trị của cột được liên kết (thường là khóa chính ở bảng kia )
  + Check : dùng để đảm bảo giá trị trong cột phải thỏa mãn điều kiện.
  + Index : dùng để lập chỉ mục cải tiến tốc độ truy xuất dữ liệu từ cơ sở dữ liệu.
  + Auto – increment : sử dụng để tự động tăng giá trị của 1 trường trong bảng. Thường là ID.

7. Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?.

* Khóa chính là trường dữ liệu (cột) được dùng để xác định mỗi record trong bảng là duy nhât từ đó giúp chúng ta phân biệt các record với nhau. Mỗi bảng chỉ nên tồn tại 1 khóa chính và khóa chính không được phép null.
* Khóa chính có thể được tạo thành từ một hoặc nhiều trường dữ liệu (cột) khóa là khóa phức hợp (composition key) được sinh ra từ mối quan hệ nhiều nhiều.

8. Khóa ngoại (foreign key) là gì?

Khóa ngoại (foreign key) là trường (cột) dữ liệu chứa các giá trị tham chiếu (trỏ đến) một khóa chính của bảng khác. Việc tạo ra các cột (trường) là khóa ngoại sẽ hình thành nên mối quan hệ giữa các bảng.

9. Phân biệt DELETE và TRUNCATE?

|  |  |
| --- | --- |
| Truncate | Delete |
| * Giống nhau là đều được xử dụng để xóa dữ liệu ở bảng | |
| * Xóa tất cả dữ liệu trong bảng. * Với bảng có khóa khóa chính đc khai báo int và auto-increment , sau khi dùng truncate xóa thì id sẽ được reset về 1. | * Có thể xóa 1 hoặc nhiều dòng hoặc tất cả. * Với bảng có khóa khóa chính đc khai báo int và auto-increment , sau khi dùng delete xóa thì id sẽ = id cũ + 1. |

10. Làm sao để truy vấn dữ liệu từ nhiều bảng?

* Để truy vấn từ nhiều bản dùng Join.

11. Các loại JOIN? Phân biệt?

* Có 2 loại chính là :
  + Inner join
  + Outer join :
    - Left join
    - Right join
    - Full join
* Phân biệt :
  + Inner Join : chỉ lấy phần chung của 2 bảng.
  + Left join : lấy phần chung của 2 bảng và phần riêng của bảng bên trái.
  + Right join : lấy phần chung của 2 bảng và phần riêng của bảng bên phải.
  + Full join là kết hợp giữa inner join + left join + right join. Với mySQL thì dùng Union kết hợp left join và right join để thay thế .

12.Phân biệt AND và OR ?

|  |  |
| --- | --- |
| AND | OR |
| * Cả 2 đều là toán tử logic dùng để kết hợp 2 hay nhiều biểu thức boolean | |
| * Trả về true khi tất cả các biểu thức boolean tham gia đều có kết quả true. Còn lại sẽ trả về false | * Trả về false khi tất cả các biểu thức boolean tham gia đều trả về false. Còn lại trả về true. |

13. Phân biệt GROUP BY và ORDER BY?

* Group by sử dụng để gộp các record theo column. Tức là các record có giá trị ở column đó mà giống nhau thì sẽ được gộp lại thành 1 nhóm .
* Order by sử dụng để sắp xếp thứ tự trong 1 column mặc định là tăng dần (ascending) giảm dần thì sử dụng thêm từ khóa descending.

14. Muốn nối chuỗi trong MySQL thì sử dụng gì?

* Sử dụng concat để nối chuỗi.

15. Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?

* Dùng hàm AVG() cho cột điểm

16. Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào?

* Sẽ group by ID cho toàn bảng và dùng hàm AVG() cho cột điểm.

17. 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 tên , avg(điểm) from table\_sinhvien group by id having avg(diem) > 5.

18. Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID?

* Dùng inner join 2 bảng student và class

19. Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?

* Dùng order by cho cột điểm .

20. 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?

* Order by ten desc, tuoi desc.

21. Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?

* Where ten like Le% and ten like %Hai.

select \* from employee where (employee\_name regexp "Nam$") and (employee\_name regexp "^Le");

22. Hiển thị các sinh viên có tên bắt đầu là ‘C’?

* Where ten regexp ‘^C’

SELECT name FROM nhanvien WHERE ten REGEXP '^Ng';

23. Phân biệt JOIN và UNION?

|  |  |
| --- | --- |
| JOIN | UNION |
| * Cho phép kết nối 2 dữ liệu trong 2 bảng khác nhau với điều kiện bảng này có khóa tham chiếu đến bảng kia. | * Cho phép hợp dữ liệu 2 câu truy vấn với điều kiện là số cột phải bằng nhau |

24. Phân biệt UNION và UNION ALL?

|  |  |
| --- | --- |
| UNION | UNION ALL |
| * Đều sử dụng để hợp dữ liệu ở 2 câu truy vấn | |
| * Loại bỏ các record trùng lặp | * Không loại bỏ |

25. Phân biệt EXIST và IN?

|  |  |
| --- | --- |
| EXIST | IN |
| * Đều sử dụng để kết hợp với truy vấn con (subquery) | |
| * Về cấu trúc thì exist không cần ghi rõ ràng column nhưng phải có điều kiện where nối ở truy vấn con và truy vấn cha * EXISTSnhanh hơn nhiều so với INkhi kết quả truy vấn con rất lớn. * vì khi dùng exist từng record ở truy vấn con sẽ đc so sánh với truy vấn cha nếu sai thì bị loại ngay. Thích hợp khi truy vấn với bảng lớn | * Phải ghi rõ ràng column * Hiệu suất không tốt với dữ liệu lớn vì khi dùng IN câu lệnh truy vấn con sẽ tìm ra tất cả record thỏa yêu cầu của truy vấn con rồi mới so sánh với yêu cầu truy vấn cha. * Thích hợp với bảng nhỏ. |

26. Làm sao để có thể tăng tốc độ truy vấn?

* Để tăng tốc độ truy vấn thì cần lập chỉ mục cho cột cần truy vấn.

CREATE UNIQUE INDEX index\_name

* ON table\_name(column 1, column 2,...);

27. Các dạng tham số ở SP?

* IN : giống tham trị ( mặc định )
* INOUT giống tham chiếu.
* OUT : chỉ lấy kết quả ra mà không truyền giá trị vào.

28. Phân biệt SP, Trigger, Function?

|  |  |  |
| --- | --- | --- |
| Store Procedure | Trigger | Function |
| * đều là tập hợp các câu lệnh query liên tiếp nhau để thực hiện 1 công việc nào đó | | |
| * Không có kiểu dữ liệu trả về * Có tham số truyền vào * Dùng call để gọi * Hàm có thể gọi từ thủ tục * Không Sử dụng trong câu lệnh truy vấn, where, having | * Không có KDL trả về * Không có tham số truyền vào * Tự động chạy khi có 1 sự kiện xảy ra trên bảng | * Có KDL trả về * Có tham số truyền vào   . Thủ tục không thể gọi từ hàm   * Có thể Sử dụng trong câu lệnh truy vấn, where, having |
|  |  |  |

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

* CSS (cascading style sheets) nó là một ngôn ngữ sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi HTML , là ngôn ngữ tạo ra phong cách cho trang web.
* Các cách nhúng :
  + Cách 1 : inline sử dụng thuộc tính style trực tiếp trong thẻ html.
  + Cách 2 : internal dùng thẻ <style> bên trong thẻ <head> của html.
  + Cách 3 : external dùng thẻ <link> bên trong thẻ <head> để nhúng 1 file css.
* Độ ưu tiên : từ gần nhất tới xa nhất. Có nghĩa là đầu tiên sẽ là inline sau đó tới internal hoặc external tùy theo vị trí được khai báo của thẻ <style> , <link>.

31. Phân biệt giữa bộ chọn class và bộ chọn id?

|  |  |
| --- | --- |
| Class | Id |
| * Đều là bô chọn (selector) trong css. | |
| * Cách khai báo .tên class. * Có thể được sử dụng để thay đổi định dạng, phong cách cho nhiều thẻ HTML khác nhau vì class không phải là duy nhất. * Độ ưu tiên thấp hơn ID | * Cách khai báo #id. * Chỉ có thể thay đổi cho 1 thẻ HTML duy nhất có Id đó. Vì ID là duy nhất. * Độ ưu tiên cao hơn class |

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

* Gồm 4 thành phần theo thứ tự từ trong ra ngoài :
  1. Content : là phần nội dung của thẻ HTML đó.
  2. Padding là khoảng cách từ content đến border.
  3. Border : là đường viền của content.
  4. Margin: là khoảng cách từ border của model này tới border của model khác.

33. Trình bày thuộc tính float, clear?

|  |  |
| --- | --- |
| FLOAT | CLEAR |
| * Thuộc tính float : sử dụng để tùy chỉnh cách thức trôi nổi của 1 phần tử trên trang web. Nó có thể nhận 4 giá trị :   1. None : Phần tử sẽ không trôi đi đâu cả ( hiển thị mặc định)   2. Left : phần tử sẽ trôi về bên trái.   3. Right : phần tử sẽ trôi về bên phải.   4. Inherit : kế thừa thuộc tính từ thành phần cha. * Khi 1 thành phần được thiết lập thuộc tính float với giá trị left hoặc right nó sẽ trôi lên hàng trên và trôi về phía bên trái hoặc bên phải khi thỏa mãn 2 điều kiện :  1. Phần tử phía trước nó cũng phải khai báo thuộc tính float left hoặc right. 2. Hàng trên nó phải đủ chiều rộng để chứa nó .  * Nếu 1 phần tử được thiêt lập float left hoặc right thì tất cả các thẻ cùng cấp khác theo sau nó sẽ được đẩy lên để lấp đầy hàng chưa phần tử đó | * Sử dụng để tùy chỉnh việc nó có bị ảnh hưởng bởi phần tử được thiết lập thuộc tính float trước nó không * Có thể nhận các giá trị như :  1. None :bị ảnh hưởng bởi thuộc tính float của thẻ trước nó. 2. Left : không bị ảnh hưởng bởi float-left. 3. Right : không bị ảnh hưởng bởi float-right. 4. Both : không bị ảnh hưởng bởi bất kì giá trị float nào trước nó. |

34. "Thuộc tính position dùng để làm gì ?

* Sử dụng để đặt các phần tử HTML ở vị trí mong muốn.

35. Phân biệt Static, absolute, relative, fixed , sticky ?

* Static : phần tử HMTL được đặt ở vị trí mặc định khi web được tạo ra. Luôn theo flow thông thường của trang .
* Relative :được sử dụng để phần tử được tạo ra cách bên trái ,phải,trên,dưới so với mặc định. Không làm ảnh hưởng tới layout của trang web ( hiển thị ban đầu ).
* Absolute : được sử dụng để phân tử được tạo ra cách bên trái phải trên dưới so với thẻ cha của nó ( thường thẻ cha được thiết lập relative ). Và làm ảnh hưởng tới layout của trang web.
* Fixed : được xét tuyệt đối so với màn hình web. Dù scroll đi đâu thì vẫn giữ vị trí đó.
* Sticky : khi scroll web tới 1 vị trí nhất định thì phần tử được thiết lập sticky sẽ được cố định trên trang web.

36. Responsive Web Design là gì?

* Là thiết kế và phát triển trang web tương thích với mọi thiết bị và môi trường của người dùng .

37. Làm thế nào để tạo 1 trang web responsive?

* Khai báo @media để chỉnh sửa cho từng column theo kích thước màn hình.

38. Bootstrap là gì?

* Là 1 framework giúp chúng ta xây dựng giao diện web 1 cách nhanh chóng và dễ dàng.

39. Một số thành phần thông dụng ở Bootstrap?

Button là các nút bạn bày ra trong trang để user nhắp vào. Nút được tạo ra với class btn và class quy định màu của nút.

* Card : bộ chứa nội dung có cấu trúc và có thể mở rộng. Bao gồm tùy chọn header   
  body footer.
* Carousel : giống như 1 slide trình chiếu . nó hiển thị nội dung xoay vòng ví dụ như ảnh ,văn bản.
* Modal : dùng khi muốn hiển thị 1 pop-up hoặc 1 dialog trên web.
* Navbar : là 1 phần của giao diện , giống như 1 thanh điều hướng để nhảy tới các page khác nhau trong web.

40. Phân biệt Static web và Dynamic web?

|  |  |
| --- | --- |
| Static web | Dynamic web |
| * Nội dung không thay đổi . * Được viết = HTML CSS JS. Không sử dụng ngôn ngữ lập trình như java, php… hoặc thiết kế DB. * Tốc độ truy cập nhanh. | * Nội dung thay đổi theo ngữ cảnh , thời gian , yêu cầu của người dùng. * Được viết bao gồm của FE ( HTML CSS JS ) và BE ( JAVA , PHP , … ) và DB. * Tốc độ truy cập chậm hơn |

41. Phân biệt 2 phương thức GET/POST?

|  |  |
| --- | --- |
| GET | POST |
| * Đều là phương thức chuyển dữ liệu từ client về server | |
| * Thông tin gửi đi được hiển thị trên URL * Bị han chế kích thước dữ liệu gửi đi. * Không thể gửi hình ảnh , video..( file nhị phân) * Không bảo mật dữ liệu * Tốc độ gửi nhanh hơn * Yêu cầu có thể được lưu trữ(cached). * Giữ lại trong lịch sử duyệt Web của trình duyệt. * Có thể được đánh dấu ghi nhớ lại(bookmarked). | * Thông tin được gửi ngầm k hiển thị trên URL. * Không bị giới hạn kích thước dữ liệu * Gửi được tất cả các file. * Bảo mật dữ liệu * Tốc độ gửi đi chậm hơn . * Yêu cầu sẽ không bao giờ được lưu trữ(cached). * Không lưu trữ lại trong lịch sử trình duyệt Web. * Không thể đánh dấu ghi nhớ lại(bookmarked). |

42. Phân biệt request/response?

* Request là yêu cầu từ client gửi về server.
* Respone là kết quả server trả về client.

43. Phân biệt forward/redirect ?

|  |  |
| --- | --- |
| Forward | ReDirect |
| * Cho phép di chuyển tới các trang nằm trong web và có thể chuyển kèm dữ liệu. | * Cho phép chuyển tới các trang khá bao gồm cả các trang nằm ngoài website. |

44. "JSP là gì? Nói: “Về bản chất JSP cũng chính là Servlet” đúng hay sai? Giải thích?

* JSP ( Java server page ) là công nghệ để phát triển các trang web động. JSP giúp các nhà phát triển java chèn java code vào trong thẻ HTML .
* Về bản chất JSP cũng chính là servlet là đúng . Vì những việc mà Jsp có thể làm thì servlet cũng vậy và ngược lại .

45. So sánh giống và khác nhau giữa JSP và Servlet?

|  |  |
| --- | --- |
| JSP | SERVLET |
| * Đều là công nghệ tạo trang web động. * Đều nằm ở phía server | |
| * Jsp mạnh về hiển thị , giao diện * Yếu về xử lí nghiệp vụ * Đóng vai trò như view. * Ngôn ngữ chính là HTML có tích hợp java | * Yếu về hiển thị , giao diện. * Mạnh về xử lí nghiệp vụ , điều phối. * Đóng vai trò như controller * Ngôn ngữ chính là java tích hợp HTML |

46.-------------------------------------------------------------------------------------

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

* JSTL ( jsp standard tag lib) : bộ thư viện hỗ trợ thẻ dựa trên ngôn ngữ java được bổ sung để tối ưu hóa lập trình JSP. Sử dụng JSTL để trang dễ đọc, dễ bảo trì, giúp tách biệt phần logic nghiệp vụ và logic trình bày .
* 1 Số thẻ :
  + <c:out > hiển thị
  + <c:forEach>
  + <c:if>
  + <c:choose> , <c:when>,<c:otherwise>

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

* MVC là mô hình kiến trúc phần mềm được tạo ra với mục đích quản lí , xây dựng dự án phần mềm có hệ thống hơn.
* Bao gồm :
  + Model : chứa các class , xử lí nghiệp vụ , thao tác với database.
  + View : Nơi chứa giao diện hiển thị , tương tác vói người dùng.
  + Controller : có nhiệm vụ điều hướng , liên kết giữa model và view.

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

* Đầu tiên là tạo kết nối với database . các thông tin kết nối cần có : địa chỉ + tên database , user name , password, nạp driver jdbc.
* Sau đó chọn 1 trong 3 cách tạo truy vấn statement , prepare statement , callable statement tùy vào trường hợp.
* Dùng resultset để lấy kết quả của câu query và dùng phương thức next() để xử lí kết quả.

50. Sự giống, khác nhau và ưu nhược điểm của MVC1 & MVC2?

Như bạn thấy trong hình trên là kiến trúc mô hình 1 như sau:

1. Trình duyệt gửi yêu cầu cho trang JSP.
2. JSP truy cập Java Bean và gọi sang business logic.
3. Java bean kết nối với cơ sở dữ liệu, nhận và lưu dữ liệu.
4. Response được gửi lại cho user bởi JSP.

|  |  |  |
| --- | --- | --- |
|  | MVC1 | MVC2 |
| Giống nhau | * Đều là mô hình kiến trúc phần mềm MVC | |
| Khác nhau | * JSP đảm nhận cả view và controller. | * Servlet đảm nhận vai trò controller * Jsp đảm nhận view |
| Ưu điểm | * Tốt cho ứng dụng nhỏ | * Tốt cho ứng dụng lớn * Tính tái sử dụng cao * Update, bảo trì thuận tiện nhanh chóng. |
| Nhược điểm | * Khó khăn trong việc update , bảo trì , mở rộng. * Khả năng tái sử dụng kém |  |

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

* Set.autoComit(false);
* Commit();
* Rollback();

52. Các tính chất của Transaction?

* Atomicity : Tất cả cùng thành công thì mới commit . chỉ cần 1 tiến trình nhỏ thất bại thì tất cả sẽ rollbak().
* Consistency : ở mọi thời điểm thì dữ liệu luôn nhất quán ( tuân theo các ràng buộc đã được định nghĩa ở database). Nếu vi phạm thì toàn bộ transaction bị hủy.
* Isolation : các transaction được bảo đảm độc lập không xung đột , dẫm chân lên nhau
* Durability : sau khi transaction đã commit(),thì dữ liệu đã cập nhật sẽ cố định dữ liệu sẽ không bị thay đổi dù cho có lỗi database.

53. Phương pháp gọi 1 SP ở JDBC?

* Tạo 1 SP trong DB .
* Dùng CallableStatement để tạo đối tượng callable kết nối với database. Xử dụng phương thức prepareCall gọi tên của SP đó .

54. Phân biệt Ant, Maven, Gradle?

|  |  |  |
| --- | --- | --- |
| ANT | MAVEN | GRADLE |
| * Ra đời năm 2000 * Không có dependency management nên phải tự tìm kiếm và import thư viện vào project | * Ra đời năm 2004 * Khắc phục lỗi của Ant.Giúp quản lí tốt hơn. Nhưng quản lí thư viện còn là file xml nên dài và rất phức tạp | * Ra đời 2009. * Quản lí thư viện được viết = groovy nên ngắn gọn hơn và hỗ trợ được nhiều dòng project với các ngôn ngữ lập trình khác nhau. |

|  |  |  |
| --- | --- | --- |
| Statement | PrepareStatement | CallableStatement |
| * Sử dụng để tạo câu lệnh truy vấn ở java. | | |
| * Tạo câu lệnh truy vấn tĩnh , không có tham số truyền vào. * Kém bảo mật vì sẽ bị nhúng SQL vào câu lệnh truy vấn . SQL injection | * Tạo câu lệnh truy vấn động bởi vì có tham số truyền vào dưới dạng dấu ? . * Bảo mật hơn | * Gọi Store Procedure. |