**BỘ CÂU HỎI PHỎNG VẤN MODULE 2**

**ADVANCED PROGRAMMING WITH JAVA**

|  |  |  |
| --- | --- | --- |
|  | **Câu hỏi:** |  |
|  | 1. *Sự khác nhau giữa JDK,JRE và JVM ?*   . JDK (Java Development Kit) giúp phát triển và biên dịch mã java thành chương trình java.  . JRE (Java Runtime Environment) giúp thực thi các chương trình java trong JMV  . JVM (Java Virtual Machine) sẽ thông dịch ra ngôn ngữ máy thực sự khi một chương trình java được thực thi. |  |
|  | 1. *Sự khác nhau giữa bộ nhớ heap và stack trong java ?*   . **Stack** là một vùng nhớ được sử dụng để lưu trữ các tham số và các biến local của phương thức mỗi khi một phương thức được gọi ra.  **. Heap** là một vùng nhớ trong bộ nhớ được sử dụng để lưu trữ các đối tượng khi từ khóa new được gọi ra, các biến static và các biến toàn cục (biến instance). |  |
|  | 1. *Trình biên dịch JIT là gì ?*   Trình biên dịch JIT (Just-In-Time) được sử dụng để tăng hiệu suất biên dịch chương trình. Trình JIT biên dịch từng phần của byte code có chức năng giống nhau tại cùng một thời điểm, do đó sẽ giảm được thời gian biên dich. Ở đây, thuật ngữ "trình biên dịch" dùng để chỉ trình dịch từ tập lệnh của một máy ảo Java (JVM) tới tập lệnh của một CPU cụ thể. |  |
|  | 1. *Platform là gì ?*   Bất cứ môi trường phần cứng hoặc phần mềm nào mà trong đó có một chương trình chạy, thì được hiểu như là một Platform. Với môi trường runtime riêng cho mình (JRE) và API, Java được gọi là Platform. |  |
|  | 1. *Sự khác nhau giữa Java platform và các platform khác ?*   Java platform khác với hầu hết các nền tảng khác theo nghĩa nó là một nền tảng dựa trên phần mềm chạy trên các nền tảng phần cứng khác. Nó có hai thành phần:   1. Môi trường thực thi (Runtime) 2. API(Application Programming Interface) |  |
|  | 1. *Tính chất "viết một lần chạy nhiều nơi" của java là gì ?*   Java code được biên dịch thành một byte code, là ngôn ngữ trung gian giữa mã nguồn và mã máy. Byte code này không phải là nền tảng cụ thể và do đó có thể được thông dịch cho bất kỳ nền tảng nào. |  |
|  | 1. *Classloader trong java là gì ?*   Classloader là một hệ thống con của JVM được sử dụng để tải các lớp và các interface. Có rất nhiều loại Classloader ví dụ: Bootstrap classloader, Extension classloader, System classloader, Plugin classloader... |  |
|  | 1. *File có tên trống ".java" có hợp lệ không ?*   Có, bạn có thể lưu file java với tên ".java", sau đó biên dịch bằng lệnh **javac .java** và chạy bằng lệnh **java ten\_lop.** |  |
|  | 1. *Các từ delete, next, main, exit và null có phải là từ khóa trong java không?*   Không. |  |
|  | 1. *Nếu không cung cấp bất kỳ đối số nào trên command line, thì mảng String của hàm main là empty hay null ?*   Mảng String là empty. |  |
|  | 1. *Chuyện gì xảy ra nếu khai báo static public void thay vì public static void?*   Chương trình được biên dịch và chạy đúng. |  |
|  | 1. *Giá trị mặc định của các biến local là gì ?*   Các biến local không được khởi tạo với bất kỳ giá trị mặc định nào, bất kể là nguyên thủy hay tham chiếu đối tượng. |  |
|  | 1. *Khác nhau Biến tham trị và Biến tham chiếu ?*   . Tham trị tức là truyền giá trị.  . Tham chiếu tức là chiếu đến 1 địa chỉ. |  |
|  | 1. *Bạn hiểu gì về Đối tượng ?*   Đối tượng là một thực thể mang tính chất vật lý cũng như mang tính logic. Đối tượng có các trạng thái và hành vi. |  |
|  | 1. *Định nghĩa Lớp (class) ?*   Lớp là một nhóm các đối tượng có các thuộc tính chung. |  |
|  | 1. *Kiểu biến nào một lớp có thể chứa ?*   Một lớp có thể gồm biến local, biến instance, và biến lớp. |  |
|  | 1. *Biến local hay biến cục bộ là gì ?tts*   . Biến local là biến được định nghĩa bên trong phương thức, constructor hoặc các khối.  . Biến cục bộ là biến sẽ được khai báo và khởi tạo bên trong phương thức và nó sẽ bị hủy khi phương thức kết thúc. |  |
|  | 1. *Biến instance là gì ?*   Biến instance là các biến bên trong một lớp nhưng bên ngoài bất cứ phương thức nào. Những biến này được thuyết minh khi lớp được tải. |  |
|  | 1. *Sự khác biệt giữa ngôn ngữ lập trình hướng đối tượng và ngôn ngữ lập trình dựa trên đối tượng là gì ?*   Ngôn ngữ lập trình dựa trên đối tượng thực hiện theo tất cả các tính năng của OOPs ngoại trừ tính kế thừa. Ví dụ về các ngôn ngữ lập trình dựa trên đối tượng là JavaScript, VBScript,... |  |
|  | 1. *Giá trị khởi tạo của biến tham chiếu đối tượng được định nghĩa là biến instance là gì ?*   Trong java, các tham chiếu của đối tượng được khởi tạo là null. |  |
|  | 1. *Constructor là gì ?*   Constructor giống như một phương thức được sử dụng để khởi tạo trạng thái của một đối tượng. Nó được gọi ra vào thời điểm tạo ra đối tượng. |  |
|  | 1. *Mục đích của constructor là gì ?*   Cung cấp các giá trị mặc định cho các đối tượng. |  |
|  | 1. *Constructor trả về kiểu giá trị gì ?*   Trả về thể hiện của lớp hiện tại. |  |
|  | 1. *Constructor được kế thừa không ?*   Không |  |
|  | 1. *Có thể tạo constructor final không ?*   Không. Constructor không thể là final. |  |
|  | 1. *Biến static là gì ?*  * Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối * tượng (mà không là duy nhất cho mỗi đối tượng), ví dụ như tên công ty của nhân viên, tên trường học của các sinh viên, … * Biến static lấy bộ nhớ chỉ một lần trong Class Area tại thời gian tải lớp đó. |  |
|  | 1. *Phương thức static là gì ?*  * Một phương thức static thuộc lớp chứ không phải đối tượng của lớp. * Một phương thức static gọi mà không cần tạo một instance của một lớp. * Phương thức static có thể truy cập biến static và có thể thay đổi giá trị của nó. |  |

|  |  |
| --- | --- |
|  | 1. *Tại sao phương thức main là static ?*   Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static. Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ bộ nhớ phụ. |
|  | 1. *Khối static là gì ?*  * Được sử dụng để khởi tạo thành viên dữ liệu static. * Nó được thực thi trước phương thức main tại lúc tải lớp. |
|  | 1. *Chúng ta có thể thực thi một chương trình không có phương thức main() không ?*   Có. Sử dụng khối static. |
|  | 1. *Chuyện gì xảy ra khi phương thức main không có static ?*   Chương trình được biên dịch, nhưng gặp lỗi. |
|  | 1. *Sự khác nhau giữa phương thức static và phương thức instance ?*  |  |  | | --- | --- | | Phương thức static | Phương thức instance | | . Một phương thức được khai báo với từ khóa static được gọi là phương thức static.  . Không cần tạo đối tượng cũng gọi được phương thức static thông qua class.  . Biến non-static không được truy cập trực tiếp trong phương thức static (hoặc khối static). | . Một phương thức không được khai báo với từ khóa static được gọi là phương thức instance.  . Phải tạo đối tượng để gọi phương thức instance.  . Biến static và non-static được truy cập trực tiếp trong phương thức instance. | |
|  | 1. *this trong java là gì ?*   Tham chiếu đến đối tượng hiện tại. |
|  | 1. *Kế thừa là gì ?*   Kế thừa là một cơ chế trong đó một lớp được thừa hưởng tất cả thuộc tính và phương thức của đối tượng khác của lớp khác. Nó được sử dụng để tái sử dụng code và ghi đè phương thức. |
|  | 1. *Lớp nào là lớp cha cho tất cả các lớp ?*   Lớp Object. |
|  | 1. *Tại sao đa kế thừa không được hỗ trợ trong java ?*   Để giảm thiểu sự phức tạp và đơn giản hóa ngôn ngữ. |
|  | 1. *Composition là gì ?*   Khai báo biến tham chiếu của một class trong một class khác được gọi là composition(sự hợp thành). |
|  | 1. *Sự khác nhau giữa aggregation và composition ?*   . Aggregation biểu diễn mối quan hệ yếu.  . Composition biểu diễn quan hệ chặt chẽ. |
|  | 1. *Tại sao java không support con trỏ ?*   Con trỏ là một biến tham chiếu tới địa chỉ ô nhớ. Nó không được sử dụng trong java vì nó không an toàn và phức tạp. |
|  | 1. *super trong java là gì ?*   Là từ khóa mà tham chiếu trực tiếp đến đối tượng của lớp cha. |
|  | 1. *Có thể sử dụng cả this() và super() trong một constructor ?*   Không, vì this() gọi đến một constructor khác trong lớp hiện tại, còn super() gọi constructor của lớp cha. |
|  | 1. *Object cloning là gì ?*   Để tạo ra một bản sao giống hệt đối tượng ban đầu. |
|  | 1. *Overloading (nạp chồng) phương thức là gì ?*   Nếu một lớp có nhiều phương thức có tên giống nhau nhưng các tham số khác nhau, được gọi là overloading phương thức (nạp chồng phương thức). Nó giúp code rõ ràng, dễ hiểu hơn. |
|  | 1. *Tại sao overloading phương thức không xảy ra khi thay đổi kiểu giá trị trả về ?*   Bởi vì đó là sự không rõ ràng, không biết gọi phương thức nào khi thực thi. |
|  | 1. *Có thể overload phương thức main() không ?*   Có. |
|  | 1. *Ghi đè (overriding) phương thức là gì ?*   Nếu lớp con có phương thức giống lớp cha được gọi là ghi đè (overriding) phương thức trong java. |
|  | 1. *Có thể ghi đè phương thức static không ?*   Không. Vì chúng thuộc về class chứ không thuộc về đối tượng. |
|  | 1. *Tại sao không thể ghi đè phương thức static ?*   Bởi vì phương thức static là một phần của lớp và nó bị ràng buộc với lớp. |
|  | 1. *Có thể ghi đè phương thức đã nạp chồng ?*   Có. |
|  | 1. *Có thể ghi đè biến instance không ?*   Không. |
|  | 1. *Sự khác nhau giữa nạp chồng và ghi đè là gì ?*  |  |  | | --- | --- | | Nạp chồng phương thức | Ghi đè phương thức | | . code dễ đọc hơn.  . Thực hiện bên trong một class.  . Tham số phải khác nhau.  . Lúc biên dịch.  . Gía trị trả về có thể giống nhau hoặc khác nhau. | . Cài đặt cụ thể cho phương thức khai báo ở lớp cha.  . Thực hiện trong 2 class có quan hệ kế thừa.  . Tham số pahir giống nhau.  . Lúc chạy chương trình.  . Gía trị trả về phải giống nhau. | |
|  | 1. *Biến final là gì* ?   Không thể thay đổi được giá trị của biến final (hằng số). |
|  | 1. *Phương thức final là gì ?*   Không thể được ghi đè. |
|  | 1. *Lớp final là gì ?*   Không thể được kế thừa. |
|  | 1. *Biến final blank là gì ?*   Một biến final không được khởi tạo giá trị lúc khai báo. |
|  | 1. *Có thể khởi tạo giá trị cho biến final blank không ?*   Có, nếu biến đó là non-static thì chỉ khởi tạo được trong constructor. Nếu biến đó là static thì chỉ khởi tạo được trong khối static. |
|  | 1. *Có thể khai báo phương thức main là final không ?*   Có, giống như: public static final void main(String[] args){} |
|  | 1. *Đa hình tại runtime là gì ?*   Đa hình tại runtime là quá trình gọi phương thức đã được ghi đè trong thời gian thực thi chương trình. Trong quá trình này, một phương thức được ghi đè được gọi thông qua biến tham chiếu của một lớp cha. |
|  | 1. *Có thể thực hiện đa hình lúc runtime với các thành viên dữ liệu không?*   Không. |
|  | 1. *Sự khác nhau giữa ràng buộc tĩnh và ràng buộc động là gì ?*   . Kiểu ràng buộc tĩnh của đối tượng được xác định tại lúc biên dịch.  . kiểu ràng buộc động của đối tượng được xác định tại runtime. |
|  | 1. *Trừu tượng là gì ?*   Tính trừu tượng là một tiến trình ẩn các cài đặt chi tiết và chỉ hiển thị tính năng tới người dùng. |
|  | 1. *Sự khác nhau giữa trừu tượng và đóng gói là gì ?*   Trừu tượng là ẩn đi cài đặt chi tiết còn đóng gói là gói code và data vào một khối duy nhất. |
|  | 1. *Lớp trừu tượng là gì ?*   Một lớp được khai báo với từ khóa abstract là lớp trừu tượng trong Java. Cần có một lớp khác kế thừa nó và cài đặt phương thức của nó. Nó không thể là thể hiện cụ thể. |
|  | 1. *Có phương thức trừu tượng không nằm trong lớp trừu tượng không ?*   Không, nếu có bất kỳ phương thức trừu tượng nào trong lớp, thì lớp đó phải là lớp trừu tượng. |
|  | 1. *Có thể sử dụng cả abstract và final cho một phương thức không ?*   Không, vì phuong thức trừu tượng (abstract) cần phải được ghi đè, trong khi đó không thể ghi đè được phương thức final. |
|  | 1. *Có thể tạo thể hiện của lớp trừu tượng không ?*   Không, lớp trừu tượng không có thể hiện. |
|  | 1. *Interface là gì ?*   Một Interface trong Java là một bản thiết kế của một lớp. Nó chỉ có các phương thức trừu tượng. Interface là một kỹ thuật để thu được tình trừu tượng hoàn toàn và đa kế thừa trong Java. |
|  | 1. *Có thể khai báo một phương thức của interface với từ khóa static không ?*   Không. Vì mặc định các phương thức của một interface là trừu tượng, từ khóa static và abtract không thể được sử dụng chung với nhau. |
|  | 1. *Một interface có thể là final không ?*   Không. Vì cài đặt phải được cung cấp bởi một lớp khác. |
|  | 1. *Marker interface là gì ?*   Một giao tiếp (interface) không có thành viên dữ liệu và phương thức được biết đến như là một giao tiếp marker.Ví dụ Serializable, Cloneable, ... |
|  | 1. *Sự khác nhau giữa lớp abstract và interface là gì ?*  |  |  | | --- | --- | | **Lớp Abstract** | **Interface** | | 1)Một lớp abstract có thể có phương thức có nội dung (các phương thức tường minh). | Interface chỉ có các phương thức  trừu tượng. | | 2)Một lớp abstract có thể có các biến instance. | Một interface không thể có các  biến instance. | | 3)Một lớp abstract có thể có constructor. | Một interface không thể có  constructor. | | 4)Một lớp abstract có thể có các phương thức static. | Một interface không thể có các  phương thức static. | | 5) Một lớp chỉ có thể extends một lớp abstract. | Một lớp thể implement  nhiều interface. | |
|  | 1. *Có thể định nghĩa private hoặc protected cho các biến trong interface không ?*   Không. chúng phải là public. |
|  | 1. *Khi nào một tham chiếu đối tượng có thể được ép sang kiểu interface tham chiếu ?*   Khi đối tượng đó implements interface được tham chiếu. |
|  | 1. *Package là gì ?*   Là một nhóm các kiểu tương tự của các lớp, giao diện và các package con |
|  | 1. *Có cần import package import java.lang không ? tại sao ?*   Không. Vì nó được tải gầm bởi JVM. |
|  | 1. *Có thể import package/lớp giống nhau hai lần không ? JVM sẽ tải package hai lần khi chạy không ?*   Chúng ta có thể import package/lớp giống nhau nhiều lần, trình biên dịch và JVM sẽ không phàn nàn gì về vấn đề này. Nhưng JVM sẽ tải lớp đó một lần. |
|  | 1. *Static import là gì ?*   Với static import, chúng ta có thể truy cập đến các thành viên của một lớp một cách trực tiếp. Không cần phải sử dụng tên class cũng có thể truy cập. |
|  | 1. *Xử lý ngoại lệ (handling exception) là gì ?*   Là một cơ chế mạnh mẽ để xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng. |
|  | 1. *Sự khác biệt giữa checked exception và unchecked exception là gì ?*   . Các lớp được extends lớp Throwable ngoại trừ RuntimeException và Error được gọi là các checked exception.  . Các lớp được extends lớp RuntimeException được gọi là unchecked exception. |
|  | 1. *Có phải mỗi khối try phải đi kèm với một khối catch ?*   Nó phải được theo sau bởi một khối [catch](https://viettuts.vn/exception-handling/khoi-lenh-try-catch-trong-java) hoặc một khối [finally](https://viettuts.vn/exception-handling/khoi-lenh-finally-trong-java). Và bất kỳ trường hợp ngoại lệ có thể bị ném sẽ được khai báo với [từ khóa throws](https://viettuts.vn/exception-handling/tu-khoa-throws-trong-java) của phương thức. |
|  | 1. *Khối finally là gì ?*  * Khối lệnh finally trong java được sử dụng để thực thi các lệnh quan trọng như đóng kết nối, đóng cá stream,… * Khối lệnh finally trong java luôn được thực thi cho dù có ngoại lệ xảy ra hay không. * Khối lệnh finally trong java được khai báo sau khối lệnh try hoặc sau khối lệnh catch. |
|  | 1. *Khối finally có thể được sử dụng mà không cần khối catch không ?*   Có. Bởi khối try. Khối finally phải theo sau khối try hoặc catch. |
|  | 1. *Có trường hợp nào khối finally không được thực thi không ?*   Khối finally không được thực thi nếu chương trình bị thoát (bằng cách gọi System.exit() hoặc lỗi phần cứng. |
|  | 1. *Sự khác nhau giữa throw và throws là gì ?*  |  |  | | --- | --- | | Throw | Throws | | . Ném ra một ngoại lệ rõ ràng.  . Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw.  . Sau throw là một instance.  . Throw được sử dụng trong phương thức.  . Bạn không thể throw nhiều exceptions. | . Để khai báo một ngoại lệ.  . Ngoại lệ checked được truyền ra ngay cả khi chỉ sử dụng từ khóa throws.  . Sau throws là một hoặc nhiều class.  . Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức.  . Bạn có thể khai báo nhiều exceptions. | |
|  | 1. *Có thể khai báo phương thức overriding của lớp con một ngoại lệ nếu phương thức của lớp cha không throw một ngoại lệ ?*   Được nhưng chỉ đối với unchecked exception còn checked exception thì không. |
|  | 1. *Việc tuyên truyền ngoại lệ là gì ?*   Là việc chuyển tiếp đối tượng ngoại lệ sang phương thức. |
|  | 1. *Có vấn đề gì không với thứ tự của các lệnh catch được viết cho FileNotFoundException và IOException ?*   Có. FileNotFoundException được kế thừa từ IOException. Các lớp con của Exception phải được bắt đầu tiên. |
|  | 1. *Điểm khác nhau giữa Error và Exception ?*   - **Error** là lỗi của chương trình.  - **Exception** là lỗi chương trình được phát hiện và có thể xử lý. |
|  | 1. *Ý nghĩa của immutable (bất biến) trong String là gì ?*   Ý nghĩa đơn giản của immutable là không thể sửa đổi hoặc không thể thay đổi được. Một khi đối tượng String đã được tạo ra, giá trị của nó không thể thay đổi. |
|  | 1. *Tại sao các đối tượng String trong java là immutable ?*   Bởi vì java sử dụng khái niệm về string literal. Giả sử có 5 biến tham chiếu, tất cả các tham chiếu đến một đối tượng "hello". Nếu một biến tham chiếu thay đổi giá trị của đối tượng, nó sẽ bị ảnh hưởng đến tất cả các biến tham chiếu khác. Đó là lý do tại sao đối tượng String trong java immutable (bất biến). |
|  | 1. *Có bao nhiêu cách để tạo ra một đối tượng String trong java ?*   Có 2 cách:  . sử dụng string literal  . sử dụng từ khóa new |
|  | 1. *Có bao nhiêu đối tượng String được tạo ra trong đoạn code sau ?*  |  | | --- | | String s1="Hello";  String s2="Hello";  String s3="World"; |   Chỉ có 2 đối tượng được tạo ra. |
|  | 1. *Tại sao java sử dụng khái niệm string literal ?*   Vì điều này khiến Java sử dụng bộ nhớ hiệu quả hơn (vì không có đối tượng String mới được tạo ra trong string constant pool). |
|  | 1. *Có bao nhiêu đối tượng được tạo ra trong đoạn code sau ?*  |  | | --- | | String s = new String("Welcome"); |   Có 2 đối tượng được tạo ra. Một đối tượng được tạo ra trong string constant pool và một đối tượng được tạo ra trong bộ nhớ heap bởi từ khóa new. |
|  | 1. *Sự khác nhau giữa String và StringBuffer là gì ?*   String là một đối tượng immutable (bất biến). StringBuffer là một đối tượng mutable (có thể biến đổi). |
|  | 1. *Sự khác nhau giữa StringBuffer và StringBuilder là gì ?*   StringBuffer là đồng bộ còn StringBuilder là KHÔNG đồng bộ. |
|  | 1. *Làm thế nào để tạo lớp immutable trong java ?*   Chúng ta có thể tạo ra lớp immutable như lớp String bằng cách định lớp với từ khóa final. |
|  | 1. *Mục đích của phương thức toString() trong java là gì ?*   Phương thức toString() trả về biểu diễn chuỗi của bất kỳ đối tượng nào. Nếu bạn in bất kỳ đối tượng, trình biên dịch java gọi phương thức toString() trên đối tượng. |
|  | 1. *Em biết gì về String Pool ?*  * String pool là một vùng nhớ đặc biệt nằm trong vùng nhớ Heap (Heap memory), dùng để lưu trữ các biến được khai báo theo kiểu String. * String pool giúp tối ưu hoá việc lưu trữ và sử dụng vùng nhớ khi khai báo biến String, giúp hạn chế tình trạng tràn bộ nhớ Java Heap Space. |
|  | *100. Nested class (lớp lồng nhau) là gì ?*  Một lớp được khai báo bên trong một lớp khác được gọi là lớp lồng nhau. Có 4 loại lớp bên trong lớp lồng nhau: member inner class, local inner class, annonymous inner class và static nested. |
|  | *101. Có sự khác nhau giữa nested class và inner class không ?*  Có. Inner class là non-static nested class nghĩa là inner class là một kiểu nested class. |
|  | *102. Nested interface là gì ?*  Bất kỳ interface nào được khai báo bên trong interface hoặc class, được biết đến nested interface. Nó có thể là static hoặc default. |
|  | *103. Có thể khai báo interface trong class không ?*  Có. Nó được biết đến như nested interface. |
|  | *104. Có thể khai báo một class trong interface không ?*  Có. Nhưng chúng không liên quan đến nhau. |
|  | *105. Sự khác nhau giữa Array và ArrayList là gì ?*   |  |  | | --- | --- | | Array | ArrayList | | . Kích thướt cố định.  . Có thể lưu trữ dữ liệu kiểu nguyên thủy và đối tượng.  . Tốc độ lưu trữ và thao tác nhanh hơn.  . Chỉ có thuộc tính length. | . Kích thước có thể thay đổi được.  . Có thể lưu trữ dữ liệu kiểu đối tượng.  . Tốc độ lưu trữ và thao tác chậm.  . Có nhiều phương thức để thao tác với dữ liệu. | |
|  | *106. Lợi thế của ArrayList so với các mảng ?*   * Kích thước của ArrayList có thể thay đổi tăng hoặc giảm tùy vào số lượng phần tử trong nó. * Thêm và xóa phần tử ở bất kì vị trí nào trong mảng một cách nhanh chóng, hiệu quả. * Có nhiều phương thức hỗ trợ các thao tác như removeAll-xóa tất cả, searching-tìm kiếm, iterations-duyệt mảng, retainAll-sửa phần tử,… * Một ArrayList có thể chứa nhiều loại đối tượng như Integer, String, Scanner, hay một đối tượng do người dùng tự khai báo. |
|  | *107. Hoạt động xóa trong LinkedList là nhanh hơn trong ArrayList, tại sao?*  Bởi vì nó không cần sắp xếp lại các phần tử sau khi thêm hoặc xóa. Nó chỉ cần cập nhật lại tham chiếu tới phần tử phía trước và sau nó. |
|  | *108. Bạn quyết định khi nào sử dụng ArrayList và LinkedList ?*  . ArrayList phù hợp với các bài toán cần thực hiện nhiều thao tác truy xuất ngẫu nhiên và ít thêm, xoá ở đầu danh sách.  . LinkedList phù hợp với các bài toán cần thêm, xoá nhiều ở đầudanh sách và ít truy xuất ngẫu nhiên. |
|  | *109. Sự khác nhau giữa ArrayList và LinkedList là gì ?*   |  |  | | --- | --- | | ArrayList | LinkedList | | . Sử dụng một mảng tự động  . Không hiểu quả với thao tác vì cần nhiều chuyển đổi  . Tốt hơn để lưu trữ và lấy dữ liệu | . Sử dụng danh sách liên kết0  . Hiệu quả cho thao tác  . Tốt hơn để thao tác dữ liệu | |
|  | *110. Sự khác nhau giữa ArrayList và Vector là gì ?*   |  |  | | --- | --- | | ArrayList | Vector | | . Không đồng bộ  . Không phải lớp kế thừa  . ArrayList tăng kích thước của nó bằng 50% kích thước mảng | . Đồng bộ  . Lớp kế thừa  . Vector tăng kích thước của nó bằng cách nhân đôi kích thước mảng | |
|  | *111. Sự khác nhau giữa Iterator và ListIterator là gì ?*   |  |  | | --- | --- | | Iterator | ListIterator | | . Duyệt các phần tử chỉ theo một chiều hướng và chuyển tiếp  . Có thể sử dụng trong List, Set và Queue | . Duyệt các phần tử theo hai hướng chuyển tiếp và ngược lại  . Có thể sử dụng trong List | |
|  | *112. Sự khác nhau giữa List và Set là gì ?*  . List: chứa các phần tử trùng lặp.  . Set: chứa các phần tử duy nhất. |
|  | *113. Sự khác nhau giữa Set và Map là gì ?*  . Set: chỉ chứa giá trị.  . Map: chứa cặp key và value. |
|  | *114. Sự khác nhau giữa Stack và Queue là gì ?*   |  |  | | --- | --- | | Stack | Queue | | . LIFO (Last in Fisrt out)  . Kết cấu: dùng một đầu để chèn và xóa các phần tử  . Số con trỏ được sử dụng: 1  . Push và Pop  . Thực hiện: đơn giản  . Biến thể: Không có biến thể. | . FIFO (Fisrt in Fisrt out)  . Có 2 đầu để xử lý dữ liệu, một đầu chèn một đầu xóa  . Số con trỏ được sử dụng: 2  . Enqueue và Dequeue  . Thực hiện: phức tạp  . Biến thể: hàng đợi tròn, hàng đợi ưu tiên và hàng đợi kết thúc gấp đôi. | |
| 2 | *115. Sự khác nhau giữa Collection và Collections là gì?*   |  |  | | --- | --- | | Collection | Collections | | . Là một Interface  . Cung cấp các chức năng về cấu trúc dữ liệu List, Set, Queue | . Là một lớp  . Sắp xếp và đồng bộ các phần tử Collection | |
|  | *116. Sự khác nhau giữa Comparable và Comparator là gì?*   |  |  | | --- | --- | | Comparable | Comparator | | . Cung cấp phương thức compareTo()  . Đặt trong java.lang package  . Lớp được sửa đổi | . Cung cấp phương thức compare()  . Đặt trong java.util package  . Lớp không bị sửa đổi | |
|  | *117. Phương thức hashCode() là gì ?*  Là phương thức được sử dụng để trả về giá trị hashcode được liên kết với object. Phương thức hashCode() thuộc class Object trong Java và có thể sử dụng nó mà không cần import thư viện nào cả. *118. Sự khác biệt giữa HashSet và HashMap là gì ?* HashSet chỉ chứa giá trị, trong khi HashMap chứa cặp key và value. *119. Sự khác nhau giữa HashMap và TreeMap là gì ?* HashMap duy trì **không có thứ tự**, trong khi TreeMap duy trì **thứ tự tăng dần.** *120. Sự khác nhau giữa HashMap và Hashtable là gì ?* - HashMap có thể chứa một khóa null và nhiều giá trị null.  - Hashtable không thể chứa bất kỳ khóa null hoặc giá trị null. |
|  | *121. Tại sao chúng ta phải ghi đè phương thức equals() ?*  Phương thức equals() được sử dụng để kiểm tra xem hai đối tượng có giống nhau hay không. Nó cần phải được ghi đè nếu chúng ta muốn kiểm tra các đối tượng dựa trên thuộc tính của chúng. |
|  | *122. Làm thế nào để đồng bộ List, Set và Map ?*  Lớp Collection cung cấp phương thức để làm cho các phần tử List, Set và Map là đồng bộ. |
|  | *123. Lợi ích của generic collection là gì ?*  Nếu chúng ta sử dụng lớp generic, chúng ta không cần typecasting. Nó là typesafe và kiểm tra tại thời gian biên dịch. |
|  | *124. Tại sao Generic được sử dụng trong Java ?*   phương pháp giúp tạo ra một phương thức mà có thể **được** gọi với nhiều kiểu dữ liệu khác nhau. |
|  | *125. Bạn hiểu gì về Access Modifier ?*  Access modifier là các từ khoá được sử dụng để quy định mức độ truy cập đến lớp và các thành phần của lớp |
|  | *126. Phân biệt public, protected, private, default ?*  • Các mức truy cập: • public: có thể truy cập từ bất cứ đâu • private:các phương thức và thuộc tính chỉ được phép truy xuất trong cùng một lớp • protected:các phương thức và thuộc tính được phép truy xuất trong cùng một lớp và ở các lớp con (kếthừa) • default:Nếu không có access modifier thì mức default sẽ được áp dụng. Lớp và các thành phần của lớp được truy xuất ở những nơi trong cùng một package. |
|  | *127. Bạn hiểu gì về Đa luồng (Multi-Thread) ?*  là một tiến trình thực hiện nhiều luồng đồng thời. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn. |
|  | *128. Phân biệt Multitasking, multiprocessing, multithreading ?*  . Multitasking: Một máy tính có thể thực thi cùng một lúc nhiều chương trình khác nhau.  .  multiprocessor là một hệ thống máy tính có từ hai bộ xử lý trở lên cùng chia sẻ bộ nhớ chính và các thiết bị ngoại vi  . Multithreading: có nhiều thread thực thi cùng lúc trong một chương trình nhầm tối ưu tài nguyên của CPU, tăng tốc độ xử lý và trải nghiệm người dùng. |
|  | *129. Thread có thể được tạo bằng hai cách nào? So sánh ?*  Cách 1: kế thừa từ lớp Thread.(Thực hiện đơn giản).  Cách 2: Impement từ Interface Runnable.(Thực hiện phức tạp). |
|  | *130. Phân biệt run(), start() ?*  . Phương thức run () cho phép chúng ta tạo một luồng mới. Sau đó chúng ta tạo một đối tượng của class vừa tạo  . Phương thức start () để bắt đầu thực thi một luồng. Start () gọi phương thức run () nằm trong đối tượng Thread. |
|  | *131. Vòng đời Thread ?*  Về cơ bản một thread có các trạng thái sau:   * New – Tạo mới * Runnable – Đang thực thi * Blocked – Bị chặn * Waiting – Trạng thái chờ * Timed Waiting – Trạng thái chờ trong một khoảng thời gian xác định * Terminated – Chấm dứt |
|  | *132. Giải thích trình dọn rác (Garbage Collector) trong Java ?*  Trong java, rác (garbage) có nghĩa là các đối tượng không còn được tham chiếu nữa.  Bộ thu gom rác (Garbage Collection) trong java được sử dụng để thực hiện quá trình tự động khôi phục lại bộ nhớ không được sử dụng tại runtime một cách tự động. Nói cách khác, đó là một cách để phá hủy các đối tượng không sử dụng nữa. |
|  | *133. Các cách mà một Thread có thể đi vào trạng thái đợi (waiting state)?*  Một Thread có thể đi vào trạng thái đợi (Waiting state) bằng việc triệu hồi phương thức sleep() của nó, bằng việc được khóa trên IO, hoặc thất bại trong việc cố gắng thu được lock của đối tượng, hoặc bởi triệu hồi phương thức wait() của đối tượng. Nó cũng có thể đi vào trạng thái đợi bởi triệu hồi phương thức suspend() của nó (phương thức này đã cũ). |
|  | *134. Cách mà Đa luồng (Multi-Thread) diễn ra trên một máy tính với một CPU đơn ?*  Scheduler của hệ điều hành cấp phát thời gian thực thi cho các Task. Bằng việc nhanh nhóng chuyển đổi giữa các Task đang thực thi, nó tạo cho chúng ta cảm tưởng rằng các Task này được thực thi đồng thời. |
|  | *135. Một Thread đã chết có thể được restart lại như thế nào ?*  Một Thread đã chết không thể restart lại được. |
|  | *136. Điểm khác nhau giữa yield và sleep ?*  Khi một tác vụ triệu hồi phương thức yield() của nó, nó chuyển thành trạng thái sẵn sàng. Khi một tác vụ triệu hồi phương thức sleep() của nó, nó chuyển sang trạnh thái đợi. |
|  | 1. *Phương thức sleep() và wait() khác nhau ở điểm nào ?*   Ví dụ, sleep(2000); làm Thread đợi đúng 2 giây. Trong khi wait(2000); làm thời gian Thread chờ có thể lên tới 2 giây. Một Thread có thể dừng việc chờ đợi sớm hơn nếu nó nhận một lời gọi notify() hoặc notifyAll().  Phương thức wait() được định nghĩa trong lớp Object và phương thức sleep() được định nghĩa trong lớp Thread. |

|  |  |
| --- | --- |
| **STT** | **Câu hỏi** |
| **A** | **SQL** |
|  | 1. **Dữ liệu (data) là gì? Phân biệt hai khái niệm dữ liệu (data) và thông tin (information) ?**   *Dữ liệu (data) là các mô tả về sự vật, hiện tượng, sự kiện… được biểu diễn dưới dạng các tài liệu chữ, số, ảnh, âm thanh, hình ảnh…*  *• Dữ liệu là rất hữu ích*  *• Việc lưu trữ dữ liệu là rất quan trọng*  *• Có nhiều cơ chế khác nhau để lưu trữ dữ liệu.* |
|  | 1. **Cơ sở dữ liệu (database) là gì ?**   *Là nhóm dữ liệu có tổ chức (ví dụ: banking(khách hàng, gửi tiền, rút tiền, chuyển khoản), school(class, student, teacher)…) được lưu trữ trên các thiết bị lưu trữ nhằm thoả mãn yêu cầu khai thác thông tin.* |
|  | 1. **Giới thiệu một số hệ quản trị cơ sở dữ liệu hiện nay ?**  * *MySQL* * *MongoDB* * *Microsoft SQL Server* * *Microsoft Asscess* * *IBM DB2* |
|  | 1. **Hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - RDBMS) là gì ?**   *CSDL Quan hệ là CSDL được xây dựng dựa trên mô hình dữ liệu quan hệ (relationalmodelofdata)*  *• Dữ liệu được tổ chức trong các bảng(table)–còn được gọi là quan hệ (relation)*  *• Các bảng bao gồm cột(column) và dòng(row)*  *• Dòng còn được gọi là bảng ghi (record) hoặc hàng(tuple)*  *• Cột còn được gọi là thuộc tính(attribute) hoặc trường(field)*  *• Một bảng thường lưu trữ dữ liệu của một loại thực thể(entity) nhất định.* |
|  | 1. **Toàn vẹn dữ liệu (data integrity) của một cơ sở dữ liệu là gì ?**   *Toàn vẹn dữ liệu (Data Integrity) của một CSDL là cách xây dựng CSDL làm sao để dữ liệu chứa trong CSDL này phải chính xác và đáng tin cậy.* |
|  | 1. **Trình bày về mô hình thực thể mối quan hệ (ERD - Entity-Relationship Diagram) ? Trong mô hình thực thể mối quan hệ có những thành phần nào ?**   ERD là một**sơ đồ**, thể hiện**các thực thể**có trong database, và**mối quan hệ giữa chúng**với nhau.  *Có 3 thành phần chính:*   * *Entity: thực thể (hoặc đối tượng) mà hệ thống quản lý.* * *Attribute : thuộc tính của các đối tượng.* * *Relationship: mối quan hệ giữa các đối tượng.* |
|  | 1. **Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL ?**   *Constraint là những quy tắc được áp dụng trên các cột dữ liệu, trên bảng. Được sử dụng để kiểm tra tính hợp lệ của dữ liệu vào, đảm bảo tính chính xác, tính toàn vẹn của dữ liệu.*  *Các loại ràng buộc trong MySQL: NOT NULL, DEFAULT, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK.* |
|  | 1. **Khóa chính (Primary key) là gì ? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?**   *Khóa chính (primary key) là trường dữ liệu (cột) có giá trị duy nhất giữa các hàng (record). Khóa chính để xác định chính xác Record nào đó trong bảng, nên khi ứng dụng hãy chọn một loại dữ liệu chuyên biệt nào đó làm khóa chính, ví dụ như mỗi người đều có một căn cước công dân khác nhau, hay mỗi bạn học viên đều có một mã học viên riêng của mình,…*   1. **Khoá 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.*  *Ví dụ: có bảng HocSinh có các trường là HOCSINHID, Ho, Ten, Ngaysinh, Dantoc, Noisinh, Diachi, GT, LopId*  *Bảng LopHoc có các trường là LopId, TenLop*  *Trong ví dụ trên LopId sẽ là khoá ngoại của bảng HocSinh vì đây là trường (cột) tham chiếu đến khoá chính của bảng LopHoc.* |
|  | 1. **Trình bày về khóa duy nhất – unique ? Phân biệt giữa Primary key và unique ?**   *UNIQUE: Sử dụng để đảm bảo dữ liệu của cột là duy nhất, không trùng lặp giá trị trên cùng 1 cột.*  *Phân biệt Primary key và Unique:*   * *Primary key: khoá chính duy nhất định danh cho mỗi hàng trong bảng, NOT NULL, tạo chỉ mục theo nhóm.* * *Unique: không phải khoá chính có nhiều hơn 1 khoá, chấp nhận NOT NULL,* tạo chỉ mục không phân cụm. |
|  | 1. **Thuộc tính not null là gì ?**   *Sử dụng để đảm bảo dữ liệu của cột không được nhận giá trị NULL* |
|  | 1. **Trình bày ràng buộc mặc định (default) ? Ví dụ ?**   *Gán giá trị mặc định cho cột trong trường hợp dữ liệu của cột không được nhập vào hay không được xác định.*  *VD: Age INT DEFAULT 22,*  *City VARCHAR (255) DEFAULT “Huế”* |
|  | 1. **Trình bày ràng buộc kiểm tra (check) ? Ví dụ ?**   *Bảo đảm tất cả giá trị trong cột thỏa mãn điều kiện nào đó. Đây là hình thức sử dụng phổ biến để kiểm tra tính hợp lệ của dữ liệu (validate data).*  *VD:* CONSTRAINT Hello300 CHECK (Age > 18 OR City="Cần Thơ") |
|  | 1. **Trình bày cách tự động gán và tăng giá trị trong 1 cột của MySQL ?**   *Thuộc tính AUTO\_INCREMENT thường được dùng trên những cột là khóa chính & có kiểu dữ liệu là số nguyên, chức năng của nó là tự động gán giá trị cho cột khi chúng ta thêm hàng mới vào bên trong bảng (giá trị của hàng đầu tiên được thêm vào bảng là 1, giá trị này sẽ tự động tăng lên một sau mỗi hàng được thêm vào bảng).* |
|  | 1. **Yêu cầu nhập vào trường tuổi của nhân viên phải từ 18-60, sử dụng ràng buộc nào ? Viết ràng buộc ?**   *CHECK*  *VD: CONSTRAINT Employee CHECK (Age > 18 AND Age < 60)* |
|  | 1. **Yêu cầu trường cặp giá trị (MaSinhVien,MaMonHoc) phải duy nhất, not null, sử dụng ràng buộc nào ? Viết ràng buộc ?**   PRIMARY KEY |
|  | 1. **Yêu cầu Tỉnh thành chỉ được nhập Hà Nội hoặc TP. Hồ Chí Minh, sử dụng ràng buộc nào? Viết ràng buộc?**   OR |
|  | 1. **Yêu cầu trường Họ tên không được để trống sử dụng loại ràng buộc nào ? Viết ràng buộc ?**   *NOT NULL*  *fullname VARCHAR (45) NOT NULL,* |
|  | 1. **Yêu cầu trường ClassID của bảng Student phải tham chiếu từ trường ClassID của bảng Class, sử dụng ràng buộc loại nào ? Viết ràng buộc ?**   *FOREIGN KEY*  *FOREIGN KEY (classId) REFERENCES class (classId)* |
|  | 1. **Yêu cầu mã sinh viên bắt đầu là chữ ‘C’, độ dài 3 ký tự, sử dụng ràng buộc nào? Viết ràng buộc?**   *SELECT studentID, studentName FROM student*  *WHERE studentName LIKE ‘H\_%’;* |
|  | 1. **Yêu cầu ngày bắt đầu làm việc phải từ ngày hiện tại trở đi, sử dụng ràng buộc nào ? Viết ràng buộc ?** |
|  | 1. **SQL là gì?**   *• SQL là viết tắt của Structured Query Language (ngôn ngữ truy vấn có cấu trúc)*  *• Được sử dụng để thao tác với các CSDL Quan hệ*  *• Có nhiều phiên bản SQL khác nhau, mặc dù hầu hết đều sử dụng chung một cú pháp, tuy nhiên cũng có một số khác biệt*  *• Chẳng hạn,một số câu lệnh SQL dành cho MySQL có thể không thực thi được trên Microsoft SQLServe* |
|  | 1. **Trình bày về câu lệnh create ?**   *Để tạo database, tạo bảng, tạo thủ tục Store Procedure, tạo view* |
|  | 1. **Trình bày về câu lệnh alter ?**   *Để thay đổi thông tin giữa các cột: thêm, sửa, xoá.* |
|  | 1. **Trình bày về các kiểu dữ liệu ký tự/chuỗi trong MySQL ? Phân biệt giữa các loại ?**  * *CHAR: Chuỗi có độ dài cố định từ 1 đến 255 ký tự.* * *VARCHAR: Chuỗi có độ dài thay đổi từ 1 đến 255 ký tự.* * *BLOB hoặc TEXT: Lưu trữ một lượng lớn dữ liệu.* * *TINYBLOB hoặc TINYTEXT: Không xác định độ dài.* * *MEDIUMBLOB hoặc MEDIUMTEXT.* * *LONGBLOB hoặc LONGTEXT.* * *ENUM.* |
|  | 1. **Trình bày về các kiểu dữ liệu số trong MySQL? Phân biệt giữa các loại?**  * *TINYINT: Số nguyên có kích cở rất nhỏ(-128 đến 127).* * *MEDIUMINT: Số nguyên có kích cỡ trung bình.* * *INT: Số nguyên có kích cỡ bình thường.* * *BIGINT: Số nguyên có kích cỡ lớn.* |
|  | 1. **Trình bày các kiểu dữ liệu ngày tháng trong MySQL? Phân biệt giữa các loại?**  * *DATE: có định dạng YYYY-MM-DD.* * *DATETIME: có định dạng YYYY-MM-DD HH:MM:SS* * *TIMESTAMP: giống với DATETIME nhưng không có dấu gạch nối* * *TIME: định dạng HH:MM:SS* * *YEAR: định dạng 2 hoặc 4 chữ số* |
|  | 1. **Yêu cầu bổ sung thêm khóa chính vào một bảng đã được tạo sẵn thì làm thế nào?**   *ALTER TABLE table\_name*  *ADD PRIMARY KEY (COLUMN);* |
|  | 1. **Yêu cầu bổ sung thêm 1 cột vào bảng đã được tạo sẵn thì làm thế nào?**   *ALTER TABLE table\_name*  *ADD new\_column constraint;* |
|  | 1. **Yêu cầu xóa một ràng buộc đã được tồn tại sẵn trong bảng thì làm thế nào?**   *ALTER TABLE table\_name*  *DROP COLUMN column\_name constraint;* |
|  | 1. **Yêu cầu sửa kiểu dữ liệu của 1 cột có sẵn trong bảng thì làm thế nào?**   *ALTER TABLE table\_name*  *MODIFY COLUMN name\_column constraint;* |
|  | 1. **Trình bày về câu lệnh insert ? insert nhiều bộ giá trị ?**   *Để thêm dữ liệu mới vào trong bảng.*  *Cú pháp:*  *INSERT INTO table\_name (column1, column 2,…)*  *VALUES (values1, values2,…)* |
|  | 1. **Trình bày về câu lệnh update? Phân biệt alter và update?**   *Dùng để cập nhật dữ liệu bên trong bảng.*  *UPDATE table\_name SET column1 = value1, column2 = value2, . . . WHERE condition;*   * *ALTER: thay đổi các thuộc tính của các mqh (thêm, sửa, xoá) trong CSDL.* * *UPDATE: cập nhật lệnh sửa đổi một hoặc nhiều bản ghi trong CSDL.* |
|  | **34.Trình bày về câu lệnh delete? Phân biệt drop và delete?**  *Dùng để xóa các hàng bên trong bảng.*  *DELETE FROM table\_name WHERE condition;*   * *DROP: xoá hoàn toàn một bảng từ CSDL.* * *DELETE: xoá một hoặc tất cả các hàng từ một bảng dựa trên điều kiện.* |
|  | **35.Bạn có biết “safe update mode”? Nếu biết, hãy trình bày về cơ chế này?**  *Cảnh báo cho bạn xoá hay update dữu liệu một cột không phải là KEY để tránh rủi ro mất mát DL.* |
|  | **36. Trình bày về câu lệnh select?**  *Dùng để truy xuất dữ liệu từ cơ sở dữ liệu.*   * *SELECT \* FROM table;* * *SELECT column1, column2…FROM table;* |
|  | **37. Trình bày về select lồng? Lấy ví dụ?**  *Là một truy vấn lồng trong truy vấn khác.*  *VD:*  SELECT SanphamID, TenSanPham, Gia FROM Sanpham  WHERE Gia > 144610.3896103896  ORDER BY Gia |
|  | **Trì38. nh bày về phép nối (join)?**  *Dùng để lấy dữ liệu từ nhiều bảng có kết nối với nhau.*  *Có 3 loại JOIN: INNER JOIN, LEFT JOIN, RIGHT JOIN* |
|  | **39. Làm sao để có thể tăng tốc độ truy vấn?**   * *Giảm thiểu sự kết nối đến hệ quản trị CSDL.* * *Chỉ SELECT những trường mà bạn cần.* * *Giải phóng bộ nhớ ngay sau khi chạy xong.* |
|  | **40. Trình bày việc sao chép dữ liệu từ bảng này sang bảng khác trong MySQL?**  *Sử dụng kết hợp câu lệnh INSERT và SELECT trong trường hợp chúng ta muốn sao chép dữ liệu từ tất cả các cột của bảng 1 sang bảng 2.*  *INSERT INTO table2 SELECT \* FROM table1 WHERE condition;* |
|  | **41. Phân biệt giữa count(\*) và count(cột)?**   * *Count(\*): trả về tổng số bảng ghi trong 1 bảng(kể cả giá trị NULL)* * *Count(column): trả về tổng số bảng ghi(bỏ qua giá trị NULL)* |
|  | **42. `Làm sao để truy vấn dữ liệu từ nhiều bảng?**  *Sử dụng JOIN* |
|  | **43. Phân biệt giữa Where và Having?**   * *WHERE: kiểm tra điều kiện trước khi tổng hợp diễn ra.* * *HAVING: kiểm tra điều kiện sau khi tổng hợp diễn ra (thường đi với lệnh GROUP BY).* |
|  | **44. Trình bày mệnh đề Order By trong câu lệnh Select?**  *Dùng kết hợp với lệnh SELECT để sắp xếp các kết quả trả về theo thứ tự tăng dần hoặc giảm dần (dựa trên giá trị của một cột nào đó)*   * *ASC: sắp xếp tăng dần.* * *DESC: sắp xếp giảm dần.* |
|  | **45.Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?** |
|  | **46. Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào?**  *SELECT S.StudentId,S.StudentName, AVG(Mark) FROM Student S join Mark M on S.StudentId = M.StudentId GROUP BY S.StudentId, S.StudentName* |
|  | **47. 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 S.StudentId,S.StudentName, AVG(Mark) FROM Student S join Mark M on S.StudentId = M.StudentId GROUP BY S.StudentId, S.StudentName HAVING AVG(Mark) > 5;* |
|  | **48. Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID)?**  *SELECT S.StudentId, S.StudentName, C.ClassName FROM Student S join Class C on S.ClassId = C.ClassID WHERE C.ClassName = 'A1';* |
|  | **Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?** |
|  | **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?** |
|  | **Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?** |
|  | **Hiển thị các sinh viên có tên bắt đầu là ‘C’?** |
|  | **53. Union là gì? Phân biệt Union và Union All?**  *UNION: Dùng để gộp kết quả của 2 hay nhiều câu lệnh truy vấn dữ liệu lại với nhau.*   * *UNION: lấy giá trị 1 lần duy nhất(không có trường hợp giá trị trùng nhau).* * *UNION ALL: lấy hết tất cả các giá trị.* |
|  | **54. Trình bày về khái niệm chỉ mục (index)? Tại sao lại cần đánh chỉ mục? Nên đánh chỉ mục trong trường hợp nào?**   * *Việc lập chỉ mục (index) trên các cột sẽ giúp cải thiện tốc độ truy vấn dữ liệu từ cơ sở dữ liệu.(tương tự như mục lục của cuốn sách)* * *Bất cứ khi nào thay đổi cấu trúc bảng dữ liệu thì đều cần Add Index lại.* * *Nên đánh chỉ mục khi: sử dụng bảng lớn; bảng có ít hoạt động update, insert; dùng chỉ mục trên các cột ít bị sửa đổi.* |
|  | **55. Có những loại chỉ mục nào? Trình bày chỉ mục duy nhất? Nếu đánh chỉ mục duy nhất lên một cột trong đó dữ liệu của cột không đảm bảo tính duy nhất thì có được không?**  *MySQL cung cấp 3 loại chỉ mục: B-Tree, Hash, R-Tree index* |
|  | **56. Trình bày về khung nhìn (view)? Phân biệt view và table?**  L*à một câu lệnh SQL được lưu trữ trong cơ sở dữ liệu với một tên xác định. Một view thực sự là một thành phần của một bảng dưới dạng truy vấn SQL được xác định trước.* |
|  | **57. Cú pháp tạo view?**  *CREATE VIEW view\_name AS*  *SELECT column1, column2…*  *FROM table\_name*  *WHERE condition;* |
|  | **58. Có thể tạo view từ view khác không?**  *Có* |
|  | **59. Trình bày về thủ tục lưu trữ (stored procedure)?**  *là chương trình con giống như một chương trình con được lưu trữ trong cơ sở dữ liệu. Trong ngữ cảnh của MySQL, nó là một phân đoạn các câu lệnh SQL khai báo được lưu trữ bên trong danh mục cơ sở dữ liệu.* |
|  | **Yêu cầu viết thủ tục lưu hiển thị họ tên sinh viên thông qua mã sinh viên là input? Chạy thủ tục với mã sinh viên là 2?** |
|  | **Yêu cầu viết thủ tục lưu đếm số lượng sinh viên có điểm thi lớn hơn một giá trị input? Chạy thủ tục với điểm là 5?** |
|  | **Viết thủ tục dựa vào số a là input, kết quả: +Nếu a lớn hơn 0 hiển thị “a là số dương” +Nếu a nhỏ hơn 0 hiển thị “a là số âm” +Nếu a bằng 0 hiển thị “a bằng 0”** |
| **B** | **SQL** |
|  | **63. Css là gì? Các cách nhúng CSS vào tài liệu html?**   * *CSS là viết tắt của cụm từ****Cascading Style Sheets*** * *CSS mô tả cách các phần tử HTML sẽ hiển thị trên màn hình, trang web, hoặc các phương tiện khác.* * *CSS giúp chúng ta tiết kiệm được rất nhiều công công sức. Nó giúp chúng ta chỉnh sửa giao diện của nhiều trang web cùng một lúc.* * *Các mô tả css có thể được lưu trong tệp tin CSS.*   *Có 3 cách để nhúng mã CSS vào trong một tài liệu HTML, bao gồm:*   1. *Khai báo trực tiếp ở thẻ HTML (Inline style sheet)* 2. *Khai báo trong thẻ <style> của tài liệu HTML (Internal style sheet)* 3. *Khai báo trong file .css riêng biệt (External style sheet)* |
|  | **64. Trình bày về các loại bộ chọn (selector) trong CSS?**   * *Bộ chọn phần tử: dựa vào****tên****của các phần tử (thẻ) HTML.* * *Bộ chọn id:  sử dụng thuộc tính****id****của một phần tử HTML để chỉ định cụ thể một phần tử HTML nào đó.* * *Bộ chọn class: chọn các phần tử có thuộc tính class với một giá trị cụ thể.* * *Gộp các bộ chọn lại với nhau: Trường hợp chúng ta có nhiều phần tử được định nghĩa kiểu giống nhau* |
|  | **65. Phân biệt giữa bộ chọn class và bộ chọn id?**   * *Bộ chọn id:  sử dụng thuộc tính****id****của một phần tử HTML để chỉ định cụ thể một phần tử HTML nào đó.* * *Bộ chọn class: chọn các phần tử có thuộc tính class với một giá trị cụ thể.* |
|  | **66. Trình bày mô hình hộp (box model) trong CSS?**  *Tất cả các phần tử HTML có thể được xem như các hộp (box). Trong CSS, thuật ngữ "mô hình hộp" (box model) được sử dụng khi nói về thiết kế và bố cục.*  *Mô hình hộp trong CSS chủ yếu là một hộp, cái dùng để bao bọc xung quanh mọi phần tử HTML. Nó bao gồm: margin, border, padding, và nội dung thực tế.* |
|  | **67. Thuộc tính margin dùng để làm gì?**  *Canh lề cho các thành phần.* |
|  | **68. Phân biệt margin, padding và border?**   * *Margin: cho phép chúng ta tạo ra các khoảng cách từ thành phần được chọn ra phía bên ngoài của thành phần đó.* * *Padding: có tác dụng căn chỉnh nội dung hiển thị trong một thành phần so với viền của thành phần đó.* * *Border: sử dụng cho các đường viền* |
|  | **69. Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed?**  *Chỉ định phương thức định vị sẽ áp dụng cho một phần tử HTML (thẻ).*   * *Static: Các phần tử HTML mặc định được định vị là "static".* * *Absolute: được định vị tương đối với phần tử ở bất cứ đâu, ngoại trừ static.* * *Relative: được định vị tương đối so với vị trí bình thường của nó.* * *Fixed: được định vị tương đối so với chế độ xem (viewport), có nghĩa là nó luôn luôn ở cùng một vị trí ngay cả khi trang được cuộn.* |
|  | **70. Trình bày thuộc tính float, clear?**   * *Float: sử dụng để chuyển một phần tử sang góc trái hoặc phải của không gian bao quanh nó, rất cần thiết trong việc định dạng bố cục trang.* * *Clear:* ***Clear****gần như là ngược lại với*float.*Thuộc tính*Clear*ngăn chặn thành phần A chiếm vùng không gian của thành phần B.* |
|  | **71. Tạo bố cục trang trong html5 với các thẻ cơ bản?** |
|  | **72. Bạn có biết Prototype? Trình bày?**  *Prototype là cơ chế mà các object trong javascript kế thừa các tính năng từ một object khác. Tất cả các object trong javascript đều có một prototype, và các object này kế thừa các thuộc tính (properties) cũng như phương thức (methods) từ prototype của mình.* |
|  | **73. Responsive Web Design là gì?**  *là kỹ thuật được sử dụng để trang web có thể hiển thị tốt trên nhiều loại thiết bị với kích thước của trình duyệt khác nhau.* |
|  | **74. Trình bày cú pháp RWD Media Queries?**  *Media Queries sử dụng @media để áp dụng một khối thuộc tính css chỉ khi thoả mãn một điều kiện nào đó.* |
| **C** | **SERVLET - JSP - JDBC** |
|  | **75. Phân biệt Static web và Dynamic web?**   * *Static web(web tĩnh): Web tĩnh tức là website cố định về nội dung, không có hoặc hạn chế về sự tương tác với website, chủ yếu sử dụng bằng mã nguồn HTML, JavaScript hoặc Flash. Nội dung cố định sau khi được thiết kế. Thường không có trang quản trị để thay đổi nội dung, cập nhật bài viết. Không có sự tương tác ngược từ người xem vào nội dung trang.* * *Dynamic web(web động):  thường có trang quản trị, để thay đổi, cập nhật nội dung trên site. Sử dụng cơ sở dữ liệu để quản lý các hạng mục trên website (bài viết, thành viên, các menu...). Người dùng có thể tương tác dễ dàng đến nội dung site. Được ứng dụng vào rất nhiều lĩnh vực : giới thiệu sản phẩm, công ty, bán hang, tin tức…* |
|  | **76. Giao thức (protocol) là gì? Kể tên một số giao thức bạn biết?**  *Protocol là một tiêu chuẩn được sử dụng để định nghĩa một phương thức trao đổi dữ liệu qua mạng máy tính, như là local area network (LAN), Internet, Intranet,… Mỗi protocol có những phương thức định dạng dữ liệu riêng khi được gửi và cần phải làm những gì một khi nó nhận dữ liệu, các dữ liệu được nén lại hay cách kiểm tra lỗi dữ liệu.* *Một số giao thức:* ***Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol over SSL/TLS (HTTPs), Internet Protocol Suite…*** |
|  | **77. HTTP là gì? Phân biệt HTTP và HTTPs?**  *HTTP (****Hypertext Transfer Protocol****)Giao thức truyền tải siêu văn bản (HTTP) là giao thức ở tầng ứng dụng (Application-level) cho các hệ thống thông tin cộng tác, phân tán, siêu dữ liệu. Đây là giao thức truyền tải dữ liệu được sử dụng để thiết lập giao tiếp giữa máy chủ cung cấp dịch vụ web và máy khách sử dụng dịch vụ web.*  *Phân biệt HTTP và HTTPs:*   * *HTTP: là văn bản có cấu trúc sủ dụng các siêu liên kết giữa các node chứa văn bản.* * *HTTPs: sử dụng với HTTP để cung cấp các dịch vụ tương tự, kết nối bảo mật được cung cấp bỡi SSL hoặc TLS.* |
|  | **78. TCP/IP là gì?**  *TCP/IP (Transmission Control Protocol/ Internet Protocol – giao thức điều khiển truyền nhận/ giao thức liên mạng) là một bộ giao thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet. TCP/IP được phát triển để mạng được tin cậy hơn cùng với khả năng phục hồi tự động.* |
|  | **79. Trình bày mô hình Client/Server?**  *Mô hình mạng Client/Server là mô hình mạng máy tính trong đó các máy tính con được đóng vai trò như các máy khách, chúng làm nhiệm vụ gửi các yêu cầu đến máy chủ, để máy chủ xử lý yêu cầu và trả kết quả về cho máy khách đó.* |
|  | **80. Máy chủ (Server) là gì?**  *Là một máy tính được kết nối với mạng máy tính hoặc internet, có IP tĩnh, có tính năng xử lý cao. Người ta cài đặt các phần mềm để phục vụ cho các máy tính khác để truy cập yêu cầu cung cấp các dịch vụ và tài nguyên.* |
|  | **81. Phân biệt máy chủ web và máy chủ ứng dụng?**   * *Máy chủ web: hữu ích phù hợp với nội dung tĩnh; tiêu thụ ít tài nguyên; sắp xếp môi trường chạy cho các ứng dụng web; đa luồng không được hỗ trợ; công suất thấp hơn máy chủ ứng dụng; giao thức HTTP và HTML được sử dụng.* * *Máy chủ ứng dụng: được trang bị cho nội dung động; sử dụng nhiều tài nguyên; sắp xếp môi trường chạy ứng dụng cho các doanh nghiệp; đa luồng được hỗ trợ; công suất cao hơn máy chủ web; giao thức HTTP và RPC/RMI được sử dụng.* |
|  | **82. Tomcat là gì? Tomcat thuộc Web Server hay Application Server?**  ***Tomcat là một web server HTTP được phát triển bởi Apache Software Foundation, hỗ trợ mạnh cho ứng dụng Java thay vì website tĩnh****. Do đó, nó có thể chạy nhiều bản Java chuyên biệt như Java Servlet, JavaServer Pages (JSP), Java EL, và WebSocket.* |
|  | **83. Phân biệt HTTP Request và HTTP Response?**   * *HTTP Request: tập hợp các thông tin gửi bởi một máy tính tới một máy chủ web.* * *HTTP Response: là dữ liệu mà server trả về cho client.* |
|  | **84. Trình bày Content Type hay MIME (Multipurpose Internet Mail Extension) Type?**  *Là tiêu chuẩn giúp phân loại các loại tập tin được sử dụng trên Internet, đặc biệt là dùng để định dạng cho thư điện tử.* |
|  | **85. Phân biệt rõ hai phương thức Get và Post?**   * *Get: số lượng dữ liệu được gửi cho mỗi request bị giới hạn được gửi qua header; không an toàn vì người dùng nhìn thấy dữ liệu rõ ràng trên URL.* * *Post: số lượng lớn dữ liệu được gửi cho mỗi request, vì nó gửi qua body; an toàn vì người dùng không nhìn thấy trên URL.* |
|  | **86. Trình bày lớp HTTPServlet?**  *HTTPServlet là một abstract class nằm trong package java.servlet.http của java servlet API. Nó extends từ một class khác tên là GeneriServlet và class GeneriServlet này đối tượng implement interface Servlet, interface chính để định nghĩa servlet của java Servlet API.* |
|  | **87. Trình bày Servlet Container?**  *Là một phần của web server được viết bởi ngôn ngữ java, tương tác với Servlet để xử lý các web động, yêu cầu từ client.*  *Servlet container thực hiện các tác vụ:*   * *Quản lý vòng đời.* * *Hỗ trợ xử lý đa luồng.* * *Object Pooling* * *Bảo mật…* |
|  | **88. Vòng đời của Servlet?**  *Là khi Servlet đó được tạo ra cho đến khi bị xoá bỏ.*  *Vòng đời Servlet được quản lý bỡi Servlet Container, có 5 giai đoạn:*   * *Nạp lớp Servlet vào bộ nhớ.* * *Khởi tạo đối tượng.* * *Gọi phương thức init() và truyền vào đối tượng ServletConfig.* * *Thực thi: gọi phương thức service() và truyền đối tượng req, resp.* * *Xoá bỏ: gọi phương thức destroy() để xoá bỏ.* |
|  | ***89. Trình bày Servlet Interface? Mô tả các phương thức init(), service(), destroy()?***  *Giao diện Servlet Interface cung cấp hành vi cơ bản của Servlet và cần phải triển khai đầy đủ trước khi có thể tạo ra bất ký Servlet nào.*  *Mô tả các phương thức:*   * *Init(): khởi chạy servlet. Nó cung cấp vòng đời và được gọi lên bởi web container chỉ 1 lần duy nhất.* * *Service(): cung cấp sự phẩn hồi tới yêu cầu được gửi đến.* * *Destroy(): gọi một lần duy nhất và chỉ ra servlet bị huỷ.* |
|  | **90. Phân biệt các loại Servlet Container: Standalone, In-process và Out-of-Servlet?**   * *Standelone: là kiểu máy chủ dựa trên java điển hình, trong đó servlet container và máy chủ web được tích hợp trong một chương trình duy nhất. vd: Tomcat* * *In-process: servlet được tách riêng ra khỏi máy chủ web.* * *Out-of-process: máy chủ web và servlet container là các chương trình khác nhau chạy trong các tiến trình riêng biệt.* |
|  | **91. Giới thiệu một số phương thức của lớp HttpServlet?**  *GET, POST, HEAD, PUT, DELETE, TRACE và OPTIONS.* |
|  | **92. Chỉ ra nhiệm vụ của hai phương thức:**  *- public void service(ServletRequest req, ServletResponse res): chuyển tiếp yêu cầu tới phương thức service() bằng cách chuyển đổi yêu cầu và phản hồi thành những đối tượng HTTP. - protected void service(HttpServletRequest req, HttpServletResponse res): nhận yêu cầu từ phương thức service và chuyển tiếp yêu cầu đó tới phương thức doGet hoặc doPost tuỳ thuộc vào phương thức của yêu cầu HTTP gưuỉ tới.* |
|  | **93. Phân biệt doGet() và doPost()?**   * *doGet(): xử lý yêu cầu GET, được sử dụng khi gửi tới một lượng nhỏ dữ liệu không nhạy cảm.* * *doPost(): xử lý yêu cầu Post, được sử dụng khi gửi tới một lượng nhỏ dữ liệu nhạy cảm. VD: id đăng nhập và mật khẩu.* |
|  | **94. RequestDispatcher được dùng làm gì?**  *Là một interface, thực hiện xác định một đối tượng có thể gửi yêu cầu tới bất kỳ tài nguyên nào (như HTML, JSP, Servlet) trên server.* |
|  | **95. Phân biệt forward() của RequestDispatcher và SendRedirect()**   * *forward(): gửi yêu cầu tương tự tới một tài nguyên khác; hoạt động ở phía server và hoạt động bên trong server.* * *SendRedirect(): luôn gửi yêu cầu mới vì nó sử dụng thanh URL của trình duyệt; hoạt động ở phía client và hoạt động bên trong và bên ngoài server.* |
|  | **96. Jsp là gì? Nói: “Về bản chất Jsp cũng chính là Servlet” đúng hay sai? Giải thích?**  *JSP (JavaServer Pages) là một ngôn ngữ kịch bản phía server, cho phép người dùng tạo ra các trang web động.*  *Đúng, vì: về bản chat thì JSP cũng chính là Servlet. Trong quá trình biên dịch JSP sẽ chuyển thành Servlet rồi Servlet này sẽ được biên dịch.* |
|  | **97. So sánh giống và khác nhau giữa Jsp và Servlet?**   * *Giống nhau: đều nằm ở phía server, đều xử lý dữ liệu động, đều chạy web container, đều là những công nghệ quan trọng trong java EE.* * *Khác nhau: JSP mạnh về xử lý giao diện, Servlet mạnh về xử lý nghiệp vụ; Servlet mã HTML nằm trong mã java, JSP mã java nằm trong mã HTML.* |
|  | **98. Vòng đời của Jsp?**  *Vòng đời của một JSP được tính từ khi JSP đó được tạo ra cho đến khi bị hủy bỏ.*  *Gồm 4 giai đoạn:*   * *Biên dịch: gồm 3 bước: phân tích trang JSP, chuyển trang JSP thành Servlet, biên dịch Servlet.* * *Khởi tạo: gọi phương thức jspinit()* * *Thực thi: gọi phương thức \_jspService()* * *Xoá bỏ: gọi phương thức jspDestroy()* |
|  | **99. Jstl là gì? Giới thiệu một số thẻ bạn đã sử dụng jstl?**  *JSP Standard Tag Library (JSTL) là thư viện thẻ chuẩn cung cấp các thẻ để kiểm soát hành vi trang, lặp đi lặp lại và các lệnh điều khiển, các thẻ quốc tế hóa, và các thẻ SQL.*  *Các thẻ cơ bản(Core Tags), các thẻ SQL(SQL Tags), các thẻ định dạng và địa phương hoá(Formartting and Localization Tags)* |
|  | **100. Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC?**  *MVC (Model – View – Controller) – MVC Pattern Design là một mô hình kiến trúc phần mềm được tạo ra với mục đích quản lí và xây dựng dự án phần mềm có hệ thống hơn, nói cụ thể thì nó là một mẫu thiết kế nhằm chia tách phần giao diện và phần code của ứng dụng để dễ dàng quản lí, bảo trì và phát triển. MVC được áp dụng trên hầu hết các ngôn ngữ lập trình hướng đối tượng hiện nay như C++, C#, Java, PHP, …*  *MVC chia ứng dụng phần mềm ra thành 3 phần có tương tác với nhau:*   * ***Model:****thường là các class chứa thông tin đối tượng (dữ liệu), tương tác truy xuất database.* * ***View:****là nơi nhận dữ liệu từ model, database và sắp xếp chúng chính xác nhờ yêu cầu từ controller và truyền về client.* * ***Controller:****Đóng vai trò trung gian giữa Model và View, nó có nhiệm vụ nhận yêu cầu từ client sau đó xử lý request, load model và gửi data qua view tương ứng với nhau rồi trả kết quả về cho client.* |
|  | **101.Phân biệt kiến trúc ba tầng (đa tầng hay n- tầng) với kiến trúc MVC?** |
|  | **102. Giới thiệu một ứng dụng tổ chức kiến trúc MVC?** |
|  | **103. Trình bày ưu và nhược điểm của kiến trúc MVC?**  *Đối với dự án nhỏ việc áp dụng mô hình MVC gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần. Phải có kiến thức vững và chuẩn, để tránh bị nhầm lẫn về các mô hình.*  *Ngược lại, nếu bạn nắm bắt tốt thì****MVC****mang lại****ưu điểm****vô cùng lớn: Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì..* |
| 104 | Sự giống, khác nhau và ưu nhược điểm của MVC1 & MVC2? |
| 105 | **API là gì?**  *API (Application programming interface - Giao diện lập trình ứng dụng) là một tài liệu có chứa mô tả về tất cả các tính năng của một sản phẩm hoặc phần mềm.* |
| 106 | **JDBC là gì? Phân biệt JDBC và ODBC?**  *Java JDBC là một java API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu. JDBC API sử dụng trình điều khiển jdbc để kết nối với cơ sở dữ liệu.*  *JDBC là viết tắt của Java Database Connectivity, nó là ứng dụng mã nguồn mở cho Java, giúp ứng dụng Java thực hiện kết nối, làm việc với CSDL.*  *Nó cho phép ta thực hiện các thao tác truy xuất, update dữ liệu với CSDL quan hệ bằng việc sử dụng các câu lệnh SQL.* |
| 107 | **Các thành phần của JDBC API? Trình bày: DriverManager, Driver, Connection, Statement, ResultSet, SQLException?**  *DriverManager – class này chịu trách nhiệm quản lý danh sách các database driver, và chịu trách nhiệm tạo kết nối tương ứng đến database cụ thể được chỉ định.*  *Driver – Interface chịu trách nhiệm xử lý các hoạt động giao tiếp giữa ứng dụng với database.*  *Connection – Interface cung cấp tất cả các hàm cho việc thao tác với database.*  *Statement – dùng các object được tạo ra từ interface này để thực thi các câu lệnh SQL xuống database.*  *ResultSet: Lưu trữ kết quả trả về từ database.*  *SQLException – Xử lý các lỗi xảy ra trong database.* |
| 108 | **Trình bày Statement? Phân biệt Statement, PreparedStatement và CallableStatement?**  *Statement – Sử dụng để thực hiện các câu truy vấn SQL tĩnh*  *PreparedStatement – Sử dụng để thực hiện các câu truy vấn SQL động hoặc có tham số*  *CallableStatement – Sử dụng để thực thi các stored procedures (Hiểu nôm na là các lệnh định nghĩa sẵn trên database)* |
| 109 | Hướng dẫn sử dụng PreparedStatement? |
| 110 | Hướng dẫn sử dụng CallableStatement? |
| 111 | stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_READ\_ONLY); Trình bày ý nghĩa các lựa chọn ResultSet ở trên? |
| 112 | Sử dụng PreparedStatement trong tính năng edit sản phẩm? |
| 113 | Sử dụng CallableStatement trong tính năng xóa sản phẩm có id cho trước? |
| 114 | Gọi MySQL Stored Procedures từ JDBC như thế nào? |
| 115 | Transaction là gì? Ví dụ? Mô tả 4 thuộc tính ACID của Transaction? |
| 116 | Những ưu điểm khi sử dụng Transaction? Cú pháp để tạo Transaction? |
| 117 | Trình bày về commit()? |
| 118 | Trình bày về hàm rollback()? |
| 119 | Trình bày về SavePoint trong Transaction? |
| 120 | Xử lý theo mẻ Batch là gì? Ví dụ? |
| 121 | Phân biệt Transaction và Batch? |

Câu 1: Ở mô hình trên DispatcherServlet là lớp nhận các yêu cầu gửi lên cũng như nhận các kết quả trả về, các bạn có thể xem DispatcherServlet trong Spring như là một Front-Controller. Kết hợp sơ đồ với giải thích bên dưới hi vọng các bạn sẽ hiểu rõ hơn.  
Khi có một yêu cầu URL gửi đến flow thực hiện như sau :

1. Spring Font Controller (DispatcherServlet) sau khi nhận yêu cầu nó sẽ tìm đến Handler Mapping thích hợp(các bạn có thể hiểu một địa chỉ(url) sẽ maping với 1 controller tương ứng).
2. Handler Mapping sẽ ánh xạ yêu cầu của client đến controller thích hợp.
3. Sau đó DispatcherServlet sẽ gửi yêu cầu đến Controller thích hợp.
4. Sau khi thực hiện tiến trình từ yêu cầu của client, nó thực thi các logic được xác định trong controller và cuối cùng trả về đối tượng ModelAndView.
5. Dựa trên các giá trị trong ModelAndView mà nó sẽ generate ra view có thể là **JSP**, **FreeMaker**, **Thymeleaf** tùy vào file cấu hình chúng ta đặt.
6. Controller sẽ gửi đến DispatcherServlet và thông qua Servlet container để trả kết quả cho client.

Các bước xử lý trong một ứng dụng web Spring MVC gồm:

1. **Front Controller** được coi là trung tâm của một ứng dụng web và cung cấp một điểm truy cập duy nhất cho tất cả các request tới. **Front Controller** gửi các request tới các bộ xử lý và cho phép các nhà phát triển sử dụng nhiều tính năng khác của framework.
2. **Front Controller** có WebApplicationContext  riêng và được kế thừa từ thư mục WebApplicationContext . Cấu hình beans trong thư mục của ứng dụng có thể được truy cập và chia sẻ giữa ngữ cảnh và thể hiện Servlet (Servlet instance) của  ứng dụng.**Front Controller**được khởi tạo trong request đầu tiên khi áp dụng cho tất cả cácServlets.
3. Một lớp được định nghĩa với annotation @Controller sẽ được xem là một Controller trong Spring MVC. Với phiên bản Spring trước 2.5 khi ánh xạ địa chỉ URL và Controller cần phải khai báo trong bộ handler. Tuy nhiên với các phiên bản mới hiện nay chỉ cần sử dụng các annotation của controller để thực hiện ánh xạ giữa địa chỉ URL và controller.
4. Interface RequestMappingHandlerMapping thực hiện tìm kiếm các annotaion @RequestMapping trong các class controllers có annotation là @Controller. Cách xử lý này có thể sử dụng để tùy chỉnh cách URL được tìm kiếm bằng cách sửa đổi các thuộc tính như interceptor, defaultHandler, order, alwaysUserFullPath, urlDecode ..v.v.v
5. Sau khi duyệt tìm các controller do người dùng định nghĩa,  controller phù hợp sẽ được chọn dựa trên ánh xạ của URL cùng với đó phương thức phù hợp sẽ được gọi. Phương thức được chọn dựa vào ánh xạ của URL và giao thức HTTP mà phương thức đó sử dụng.
6. Sau khi xử lý các logic nghiệp vụ viết trong phương thức của controller, là thời điểm để tạo ra các response (hồi đáp). Các hồi đáp này khác với HTTPRespones vì nó không trực tiếp đáp ứng người dùng. Thay vào đó response sẽ được gửi đến Front Controller.  Response ở đây có thể là tên logic của view, tên logic của data model, và dữ liệu thực tế để liên kết. Thông thường đối tượng của ModelAndView được trả về Front Controller.
7. Tên logic của view cũng như Front Controller, nhưng view này không chứa bất kì thông tin về các trang(page) thực tế trả về cho người dùng. Các thẻ bean cấu hình cho ViewResolver nằm trong file XXX-servlet.xml sẽ thực hiện ánh xạ giữa tên của view và trang (page) thực trả về cho người dùng. Có nhiều cách xử lý của view hỗ trợ bởi framework, nên ở đây chúng tôi chỉ mô tả ngắn gọn.
8. ViewResolver là bộ phận quyết định xem Front Controller sẽ trả về View thực tế nào. Front Controller sẽ xây dựng view trả về bằng cách truy xuất các giá trị đã được gắn cho dữ liệu ở Model, kết quả này sau đó sẽ được trả về cho người dùng.

Câu 2: Trong kĩ thuật phầm mềm, Dependency injection là một công nghệ mà theo đó một object cung cấp sự phụ thuộc cho một object khác.

* Dependency là một object có thể được sử dụng (xem nó như 1 service).
* Injection là việc chuyển Dependency vào object client ( object sẽ sử dụng service). Việc injection service đến client thay vì để client tìm và gọi service là nền tảng cơ bản của pattern Dependency injection.

Dependency injection là một cơ chế để triển khai IoC

• Các dependency được cung cấp và điều khiển bởi container hoặc framework

• Thao tác “tiêm” các đối tượng vào trong đối tượng khác được thực hiện bởi container hoặc framework

Câu 4: Framework là các ứng dụng phần mềm có tính trừu tượng (abstraction), cung cấp các tính năng chung và thông dụng, có thể tuỳ biến để tạo nên những ứng dụng cụ thể khác nhau

• Mỗi framework cung cấp một phương pháp riêng biệt để xây dựng và triển khai ứng dụng

• Mỗi framework bao gồm một môi trường tổng thể, tái sử dụng được nhằm cung cấp các chức năng và công cụ để hỗ trợ quá trình phát triển ứng dụng

Câu 5: Spring MVC là một framework MVC sử dụng một trình điều phối để gọi các phương thức trong controller và chuyển tiếp điều khiển đến một view (khung nhìn). Đây là lợi ích đầu tiên của việc sử dụng Spring MVC: Bạn không cần viết servlet điều phối của riêng mình. Dưới đây là danh sách các tính năng mà Spring MVC được trang bị để phát triển nhanh hơn:

SpringFrameworkcungcấpmộtmôhìnhđầyđủchoviệcpháttriển và cấu hình các hệ thống Java lớn • Các tính năng lõi: • Core: IoC container, Events, Resources, i18n, Validation, Data Binding, Type Conversion, SpEL, AOP. • Testing: Mock objects, TestContext framework, Spring MVC Test, WebTestClient. • Data Access: Transactions, DAO support, JDBC, ORM, Marshalling XML. • Web Servlet: Spring MVC, WebSocket, SockJS, STOMP messaging. • Web Reactive: Spring WebFlux, WebClient, WebSocket. • Integration: Remoting, JMS, JCA, JMX, Email, Tasks, Scheduling, Cache. • Languages: Kotlin, Groovy, Dynamic languages

Câu 6: Framework giải quyết các vấn đề thông dụng, giúp lập trình viên tập trung vào xử lý nghiệp vụ

• Giúp tăng tốc độ phát triển

• Cung cấp môi trường làm việc tiêu chuẩn, giúp dễ giao tiếp giữa các bên khi cùng tham gia phát triển

• Các framework thường có cộng đồng phát triển lớn, các giải pháp đã được đánh giá và thử nghiệm, hệ sinh thái đầy đủ giúp nhanh chóng xây dựng được các giải pháp tuỳ biến

Câu 7,8 : Inversion of Control (IoC) là một nguyên lý trong phát triển phần mềm, trong đó việc điều khiển các đối tượng hoặc các thành phần của hệ thống được thực hiện bởi framework hoặc các container

IoC cho phép framework nắm giữ quyền điều khiển luồng thực thi của hệ thống và gọi các mã nguồn khác

Lợi ích:

• Tách rời việc thực thi (execution) và việc triển khai (implementation)

• Dễ chuyển đổi giữa các implementation

• Dễ phân tách module hơn

• Dễ kiểm thử hơn, bằng cách tách rời các thành phần riêng lẻ

• Có thể triển khai IoC thông qua một số cơ chế: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection(DI)

Câu 9:  **bean là** những module chính của chương trình, được tạo ra và quản lý bởi Spring IoC container. Các **bean** có thể phụ thuộc lẫn nhau,

Spring Bean là các object trong Spring Framework, được khởi tạo thông qua Spring Container. Bất kỳ class Java POJO nào cũng có thể là Spring Bean nếu nó được cấu hình và khởi tạo thông qua container bằng việc cung cấp các thông tin cấu hình (các file config .xml, .properties..)

IoCcontainerđượcđạidiệnbởiinterfaceApplicationContext • Springcontainerchịutráchnhiệmkhởitạo,cấuhìnhvàtổchứccác đối tượng – còn được gọi là các beans • SpringMVC cungcấp một số triển khai: • ClassPathXmlApplicationContext • FileSystemXmlApplicationContext • WebApplicationContext

Câu 10:

Có 5 scope được định nghĩa cho Spring Bean:

* **Singleton:**Chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container. Đây là scope mặc định cho spring bean. Khi sử dụng scope này cần chắc chắn rằng các bean không có các biến/thuộc tính được share.
* 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.

Câu 11: @Autowired là annotation được sử dụng để tiêm các đối tượng

Câu 12: Chú thích Spring Component được sử dụng để biểu thị một lớp là Thành phần. Điều đó có nghĩa là Spring framework sẽ tự động phát hiện các lớp này để đưa vào phụ thuộc khi cấu hình dựa trên chú thích và quét classpath được sử dụng.

@Component được sử dụng để định nghĩa một khuôn mẫu chung cho toàn bộ dự án.

Class được đánh dấu với annotation @Component thì sẽ được tạo thành 1 bean. Khi một class được đánh dấu với annotation với @Component thì khi dự án Spring được chạy nó sẽ quét qua và tạo 1 bean cho class đó.

Spring chỉ tìm kiếm và chọn ra những class được đánh dấu với @Component annotaion để khởi tạo và đăng ký với ApplicationContext.

Câu 13: • Các annotation này thực hiện các nhiệm vụ như ánh xạ tới URL, khai báo tham số của request, xử lý ngoại lệ…

Spring controller nhận ủy quyền xử lý các request từ phía front controller. Công việc cơ bản của nó là nhận dữ liệu cần thiết từ front controller, xác định view cần sử dụng, triệu gọi lớp xử lý nghiệp vụ cần thiết, sắp xếp dữ liệu mà view cần đến vào model, gửi thông tin về view cùng model lại cho front controller.

Tới bản SpringMVC hiện tại, sau khi cho phép sử dụng annotion controller, một class được ấn định thành controller bằng cách sử dụng annotation @Controller.

Springcontrollerxửlýcác requestđể thực hiện các logic nghiệp vụ

Bộ điều khiển - Bộ điều khiển chứa logic nghiệp vụ của một ứng dụng. Ở đây, chú thích @Controller được sử dụng để đánh dấu lớp là bộ điều khiển. ... Front Controller - Trong Spring Web MVC, lớp DispatcherServlet hoạt động như một front controller. Nó chịu trách nhiệm quản lý luồng của ứng dụng Spring MVC.

Câu 14: Class ModelAndView có nhiệm vụ quan trọng trong việc tạo ra các hồi đáp (response). Các thể hiện của ModelAndView giúp dễ dàng tạo liên kết giữa data model và view. Trong đó data model là đối tượng chứa dữ liệu để hiển thị lên view.

Câu 15: Class ModelMap là lớp con của LinkedHashMap được sử dụng trong việc xây dựng data model bằng sử dụng kiểu dữ liệu cặp gồm key và value. ModelMap sử dụng phương thức addAttribute() để tạo liên kết giữa model và tên logic trong model. Thuộc tính đặt trong ModelMap có thể sử dụng để hiển thị dữ liệu dạng form trong sự kiện submit form.

Câu 16: Các **ViewResolver** (trình phân giải view) **là** các đối tượng chịu trách nhiệm lấy các đối tượng View cho một hành động và vị trí cụ thể. Thông thường, các controller yêu cầu ViewResolvers chuyển tiếp đến một view với một tên cụ thể (một chuỗi được trả về bởi phương thức).

**URL**

* URL được sử dụng để mô tả đặc điểm của một mục.
* URL liên kết một trang web, một thành phần của trang web hoặc một chương trình trên một trang web với sự trợ giúp của việc truy cập các phương pháp như giao thức.
* URL cung cấp thông tin chi tiết về loại giao thức sẽ được sử dụng.
* URL là một loại URI.

**URI**

* URI cung cấp một kỹ thuật để xác định danh tính của một mục.
* URI được sử dụng để phân biệt tài nguyên này với tài nguyên khác bất kể phương pháp được sử dụng.
* URI không chứa đặc tả giao thức.
* URI là "tập cha" của URL.

URI : là một chuỗi ký tự được sử dụng để xác định, nhận dạng một tên hoặc một tài nguyên.[[](https://vi.wikipedia.org/wiki/URI" \l "cite_note-1)

Controller là đối tượng nhận và xử lý các request

• Sử dụng annotation @Controller để khai báo các controller

• Sử dụng @RequestMapping để ánh xạ các URL tới các handler method củacontroller

• @RequestMapping có các biến thể khác như @GetMapping, @PostMapping,@PutMapping…

• 3 component quan trọng của Spring MVC là @Controller, @Service và @Repository

• Kiến trúc Repository là một kiến trúc tốt để xây dựng các ứng dụng web Spring MVC

Câu 21 : Annotation RequestMapping dùng để ánh xạ một request tới một phương thức. Bạn có thể sử dụng @RequestMapping chú thích cho một phương thức hoặc một lớp. Một phương thức chú thích bởi annotation  @RequestMapping sẽ trở thành phương thức xử lý request và sẽ được gọi khi dispatcher servlet nhận được một request với URL phù hợp.

Câu 22 : **Data Binding** là **cơ chế** liên kết dữ liệu đầu vào hoặc đầu ra với các đối tượng model. ... Khi tương tác với form, dữ liệu trên form sẽ được tự động chuyển đổi thành các thuộc tính của đối tượng liên kết với nó. Khi thay đổi dữ liệu của đối tượng thì dữ liệu trên các điều khiển cũng thay đổi theo

Câu 23 : Sử dụng thuộc tính consumes để ánh xạ đến Content-Type của request

Sử dụng thuộc tính produces để ánh xạ đến Accept của Request

Câu 24 : View là một trong 3 thành phần của mô hình ứng dụng MVC

• View có nhiệm vụ hiển thị dữ liệu và là giao diện giúp người dùng tương tác với ứng dụng

Các ViewResolver (trình phân giải view) là các đối tượng chịu trách nhiệm lấy các đối tượng View cho một hành động và vị trí cụ thể. Thông thường, các controller yêu cầu ViewResolvers chuyển tiếp đến một view với một tên cụ thể (một chuỗi được trả về bởi phương thức). Sau đó, tất cả các view resolver trong ứng dụng thực thi theo thứ tự cho đến khi một trong số đó có thể phân giải view trong trường hợp một đối tượng View được trả về và điều khiển được truyền tới nó cho việc render HTML.

Có hai interface trong Spring MVC để phù hợp với cốt lõi hệ thống template của nó đó là:

• org.springframework.web.servlet.View

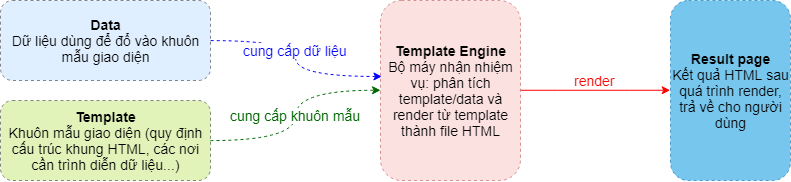
• org.springframework.web.servlet.ViewResolver

• Các view mô hình các trang trong ứng dụng và cho phép sửa đổi và định nghĩa trước các hành vi thông qua việc định nghĩa chúng như các bean.

• ViewResolvers là các đối tượng phụ trách thu thập các đối tượng View cho một hoạt động và khu vực cụ thể

•

Template là một cách thức thông dụng để hiển thị kết quả của một request về cho người dùng. Template định nghĩa một khuôn mẫu sẵn, sau đó ứng dụng sẽ đổ dữ liệu vào template để tạo ra một response thực tế và trả về cho người dùng.



Thymeleaf là một Java template engine phía máy chủ, nó có thể làm việc với cả hai môi trường, web và non-web, nó có khả năng xử lý HTML, XML, JavaScript, CSS và thậm chí cả văn bản thuần túy.

Mục tiêu chính của Thymeleaf là mang lại các template tự nhiên, đồng nhất, đơn giản (nature templates) và dễ dàng bảo trì. Để đạt được điều này, Thymeleaf xây dựng trên khái niệm về các mầu tự nhiên (natural templates) để chèn logic của nó vào các file template mà không ảnh hưởng đến template. Điều này giúp thu hẹp khoảng cách giữa các nhóm thiết kế và phát triển.

Khi sử dụng th:each, Thymeleaf cung cấp một cơ chế hữu ích cho việc theo dõi trạng thái của sự lặp lại: biến trạng thái (status variable).

Các biến trạng thái được định nghĩa trong thuộc tính th:each và chứa các dữ liệu sau:

* Chỉ số lặp hiện tại (*iteration index*), bắt đầu bằng 0. Đây là thuộc tính chỉ mục (index).
* Chỉ số lặp hiện tại(*iteration index*),  bắt đầu bằng 1. Đây là thuộc tính đếm (count).
* Tổng số phần tử trong biến lặp. Đây là thuộc tính kích thước (size).
* Biến lặp cho mỗi lần lặp. Đây là thuộc tính hiện tại (current).
* Cho dù lặp hiện tại (current iteration) là chẵn hay lẻ. Đây là những thuộc tính boolean even / odd.
* Cho dù lặp hiện tại (current iteration) là lần đầu tiên. Đây là thuộc tính boolean first.
* Cho dù lặp hiện tại (current iteration) là lần cuối cùng. Đây là thuộc tính boolean last.

Câu 26: DataBindinglàcơchếliênkếtdữliệuđầuvào(hoặcđầura)vớicác đối tượngmodel • DataBindinggiúpchoviệc tươngtác với dữliệutrởnêndễdàng • SửdụngDataBinding,cácformđềuđượcliênkếtvớimộtđốitượng biểu diễn dữ liệu ở phía sau • Khitươngtácvớiform,dữliệutrênformsẽđược tựđộngchuyểnđổi thành các thuộc tính của đối tượng liênkết với nó • Data Binding hỗ trợ chuyển đổi dữ liệu (data conversion) và validate dữ liệu • DataBindingtrongSpringhoạt độngdựatrên DataBinde

Câu 27: Formatter cũng hoạt động giống như converter, tức là chuyển đổi một kiểu dữ liệu sang kiểu dữ liệu khác. Tuy nhiên, kiểu dữ liệu nguồn của Formatter là String, trong khi đó converter có thể làm việc với bất cứ kiểu dữ liệu nguồn nào. Formatter phù hợp hơn với tầng web (web-tier), còn converter thì có thể sử dụng ở bất cứ tầng nào. Trong trường hợp muốn chuyển đổi dữ liệu nhập vào từ một trường input trong form trong một ứng dụng Spring MVC, chúng ta nên lựa chọn formatter hơn là converter.

Câu 28: ORM (Object-Relational Mapping) là kỹ thuật liên kết giữa các đối tượng trong lập trình với các đối tượng trong CSDL. Khi sử dụng lập trình hướng đối tượng và CSDL quan hệ, ORM sẽ giúp ánh xạ các lớp đến các bảng và các đối tượng đến các bản ghi trong các bảng.

**Lợi ích của ORM**

* Cho phép truy xuất dễ dàng đến dữ liệu thông qua các đối tượng lập trình
* Giúp lập trình viên tập trung thao tác với các đối tượng, không cần quá quan tâm đến CSDL thực tế đang dùng
* Tự động hoá rất nhiều thao tác như quản lý giao dịch, validate dữ liệu, cache

**Các thành phần trong kiến trúc của Hibernate**

**Persistence object**

Chính là các POJO object map với các table tương ứng của cơ sở dữ liệu quan hệ. Nó như là những container chứa dữ liệu từ ứng dụng để lưu xuống database, hay chứa dữ liệu tải lên ứng dụng từ database.

**Configuration**

Configuration là đối tượng Hibernate đầu tiên cần tạo trong bất kỳ ứng dụng Hibernate nào và chỉ cần tạo một lần trong quá trình khởi tạo ứng dụng. Nó đại diện cho một tập tin cấu hình hoặc thuộc tính yêu cầu của Hibernate. Đối tượng Configuration cung cấp hai thành phần chính:

* Database Connection: Thao tác này được xử lý thông qua một hoặc nhiều tệp cấu hình được Hibernate hỗ trợ. Các tệp này là hibernate.properties và hibernate.cfg.xml.
* Class Mapping Setup: Thành phần này tạo ra kết nối giữa các lớp Java và các bảng cơ sở dữ liệu.

**Session Factory**

Session Factory là một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong Hibernate configuration.

SessionFactory là một đối tượng luồng an toàn (Thread-safe) và được sử dụng bởi tất cả các luồng của một ứng dụng.

**Session**

Một session được sử dụng để có được một kết nối vật lý với một cơ sở dữ liệu.

Các đối tượng Session không nên được mở trong một thời gian dài bởi vì chúng thường không phải là luồng an toàn (thread-unsafe) và chúng cần được tạo ra và được đóng khi cần thiết.

Mỗi một đối tượng session được Session factory tạo ra sẽ tạo một kết nối đến database.

**Transation**

Một Transaction đại diện cho một đơn vị làm việc với cơ sở dữ liệu và hầu hết các RDBMS hỗ trợ chức năng transaction. Các transaction trong Hibernate được xử lý bởi trình quản lý transaction và transaction.

Transaction đảm bảo tính toàn vẹn của phiên làm việc với cớ sở dữ liệu. Tức là nếu có một lỗi xảy ra trong transaction thì tất cả các tác vụ thực hiện sẽ thất bại.

## **JPA – Java Persistence API**

JPA là một bộ đặc tả tiêu chuẩn của Java, trong đó mô tả các thao tác quản lý dữ liệu quan hệ trong các ứng dụng sử dụng Java.

JPA bao gồm:

* Bộ API, được định nghĩa trong gói javax.persistence
* Java Persitence Query Language (Ngôn ngữ truy vấn Java Persistence)
* Object/relational metadata

## **Entity**

Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng, chẳng hạn Customer, Product, Student… có thể là các entity.

Entity thường là POJO (Plain Old Java Object) – nghĩa là các lớp đơn giản, chỉ dùng để mô tả dữ liệu chứ không xử lý các logic nghiệp vụ.

Entity sẽ được ánh xạ (mapping) tới một bảng trong CSDL. Chẳng hạn, lớp Customer có thể ánh xạ tới bảng customers trong CSDL, lớp Student có thể ánh xạ tới bảng students trong CSDL.

Trong JPA, một entity cần tuân thủ:

* Được gắn với annotation javax.persistence.Entity
* Có một constructor là public và không có tham số
* Không được khai báo final
* Các biến đối tượng cần được khai báo là private, protected hoặc ở mức package-private

## **Persistence Context**

Persistence Context là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu.

## **Entity Manager**

Entity Manager là đối tượng quản lý các entity

Hibernate là một trong những ORM(Object Relational Mapping) Framework. Hibernate framework là một framework cho persistence layer. Như vậy, nhờ có Hibernate framework mà giờ đây khi phát triển ứng dụng thì chỉ còn chú tâm vào những layer khác mà không phải bận tâm nhiều về persistence layer.

Hibernate giúp lập trình viên Java có thể map các object (POJO) với database, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cơ dữ liệu quan hệ.

Ngoài ra Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh. Hibernate cho phép truy vẫn dữ liệu thông qua Java Persistence API (JPA) hoặc bằng HQL hoặc bằng SQL thuần (Native SQL).

Hibernate Query Language (HQL) là một ngôn ngữ truy vấn hướng đối tượng (OOP), tương tự như SQL (Structured Query Language) nhưng thay vì làm việc trên các bảng và cột, HQL làm việc với các đối tượng persistent và các thuộc tính của chúng.

Trạng thái Entity

### ****Tìm entity theo id****

### ****Lưu một entity****

### ****Cập nhật một entity****

### ****Xoá một entity****

|  |  |
| --- | --- |
| [Many-to-One](https://viettuts.vn/hibernate/hibernate-many-to-one-relationship-voi-mysql-tren-eclipse) | Mapping mối quan hệ many-to-one trong Hibernate |
| [One-to-One](https://viettuts.vn/hibernate/hibernate-one-to-one-relationship-voi-mysql-tren-eclipse) | Mapping mối quan hệ one-to-one trong Hibernate |
| [One-to-Many](https://viettuts.vn/hibernate/hibernate-one-to-many-relationship-voi-mysql-tren-eclipse) | Mapping one-to-many relationship using Hibernate |
| [Many-to-Many](https://viettuts.vn/hibernate/hibernate-many-to-many-relationship-voi-mysql-tren-eclipse) | Mapping mối quan hệ many-to-many trong Hibernate |

**Câu 37:Session Factory Là** một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong một file xml và mỗi loại Database khác nhau sẽ có một cấu hình khác nhau. Từ đó, mỗi lần cần query hay insert dữ liệu, chúng ta sẽ tạo ra Session và sử dụng

Câu 38: **detached**, **persistent** và **transient**.

Các đối tượng trở nên **detached** (tách rời) sau khi phiên được đóng bởi nhà phát triển. Các đối tượng tách rời sau đó có thể biến thành các đối tượng ổn định với các phương thức thích hợp được kích hoạt. Các đối tượng **persistent**(liên tục) là những đối tượng đang mở - chúng xảy ra bất cứ khi nào bạn lưu hoặc truy xuất một số trường hợp nhất định từ cơ sở dữ liệu Hibernate. Cá**c** đối tượng **transient** (không ổn định) là những*“đối tượng lẻ”*- vừa được tạo và chưa được phân biệt bởi một Phiên cụ thể.

Spring Data Repository giúp giảm thiểu lượng code thông thường lặp đi lặp lại ở tầng truy xuất dữ liệu. •Spring Data Repository định nghĩa một interface chính tên là Repository. Interfacenàynắmbắtentitycầnquảnlývàkiểudữliệu id của entity đó. •Interface CrudRepository kế thừa từ interface Repository, chứa các phươngthức thôngdụngdànhchocác thao tácCRUD. •Interface PagingAndSortingRepository cung cấp các phương thức hỗ trợ choviệc phân trang và sắp xếpcác entity

Câu 39:

Validation là thao tác cho phép ứng dụng kiểm tra dữ liệu nhập vào bởi người dùng để đảm bảo tính hợp lý và chính xác khi xử lý các thao tác nghiệp vụ. Chẳng hạn, khi cần nhập địa chỉ email, ứng dụng cần đảm bảo email đó là hợp lệ.

Validation có thể được thực hiện ở các tầng khác nhau, chẳng hạn:

* Validate ở tầng giao diện: Sử dụng JavaScript để validate
* Validate ở tầng back-end: Validate ở Controller/Service
* Validate ở tầng CSDL: Sử dụng các ràng buộc trong CSDL (NOT NULL, UNIQUE...)

## **Validation sử dụng interface Validator của Spring**

Spring hỗ trợ interface Validator để thực hiện các thao tác kiểm tra tính hợp lệ của dữ liệu nhập vào. Interface Validator hoạt động bằng cách sử dụng một đối tượng Errors trong khi tiến hành xác minh, đối tượng xác minh có thể báo cáo cho đối tượng Error những sai phạm trong quá trình xác minh.

JSR-303 tiêu chuẩn hóa việc khai báo ràng buộc validation và metadata cho nền tảng Java. Sử dụng API này, bạn sử dụng annotation kết hợp với các ràng buộc validation cho các thuộc tính của model và runtime sẽ thực thi chúng. Có một số ràng buộc tích hợp sẵn mà bạn có thể sử dụng. Bạn cũng có thể định nghĩa các ràng buộc của mình.

Tóm lại, đối với những thao tác validate cơ bản, chúng ta sẽ sử dụng annotation @Valid trong các lệnh gọi phương thức. Mặt khác, đối với thao tác validate theo nhóm, chúng ta sẽ cần sử dụng annotation @Validated của Spring. Các @Valid  annotation cũng cần thiết để validate các thuộc tính lồng bên trong (nested properties).

Câu 41: [BindingResult] là đối tượng của Spring mà **giữ kết quả xác thực và ràng buộc và chứa các lỗi có thể xảy ra.**BindingResult phải đến ngay sau khi đối tượng mô hình được xác thực nếu không Spring sẽ không xác thực đối tượng và ném ngoại lệ.

### Câu 42: Join points

Điểm nối, là những vị trí xác định trong luồng thực thi của chương trình mà tại đó một khía cạnh sẽ bắt đầu tham gia vào. Điểm gia nhập có thể là một lời gọi phương thức, một lệnh tung expection, hay một lệnh chỉnh sửa thuộc tính ...

### Advice

Là hành động thực sự mà aspect sẽ làm, trước, hoặc sau - khi thực thi phương thức. Đây là mô tả quan trọng nhất của một mối quan tâm - nó làm gì, và khi nào.

### Weaving

Dệt – l0à hành động ghép aspect và đối tượng mục tiêu áp dụng của nó vào với nhau thành một object mới (và sau đó chương trình thực thi theo object mới đó). Aspect có thể được dệt tại một vài thời điểm:

* Khi compile: aspect được dệt vào mục tiêu tại thời điểm compile. Điều này cần có compiler đặc thù. Framework AOC AspectJ là một framework điển hình.
* Khi nạp class mục tiêu vào JVM: điều này yêu cầu một class loader đặc thù (xem lại về cách hoạt động của JVM), framework AspectJ cung cấp một classloader điển hình.
* Tại thời điểm thực thi: dệt diễn ra khi chương trình đang chạy, đây là cách được Spring sử dụng.

### Pointcut

Điểm cắt là những biểu thức đi theo join points để xác định xem một hay một tập hợp các advice sẽ được khởi động hay không.

### Câu 43: Các loại Advice

Spring hỗ trợ 5 loại advice sau để đưa một aspect vào tiến trình thực thi

* before: được thực thi trước khi method được thực thi
* after: được thực thi sau khi method được thực thi
* after-returning: được thực thi sau khi và chỉ nếu khi method được thực thi thành công
* after-throwing: được thực thi sau khi và chỉ nếu khi method được thực thi không thành công, nghĩa là có ngoại lệ được throw
* around: được thực thi cả trước và sau khi method được thực thi

## Câu 46: **1. Phân loại các ngoại lệ**

Ngoại lệ có thể là một trong các loại sau:

* Ngoại lệ nghiệp vụ: mô tả một vi phạm về luật nghiệp vụ. Chẳng hạn chuyển một số tiền vượt quá hạn mức. Hay đặt một phòng khách sạn vào một ngày ở quá khứ.
* Ngoại lệ từ thư viện, sinh ra trong khi chương trình đang hoạt động bình thường. Chẳng hạn ngoại lệ thời gian chờ quá lâu, hay quá nhiều request cùng lúc.
* Ngoại lệ từ hệ thống. Những ngoại lệ không bao giờ xảy ra nếu hệ thống đang hoạt động bình thường. Ví dụ IOException.
* Ngoại lệ bất thường từ hệ thống. Ví dụ không có quyền ghi đọc file, hay hết dung lượng ổ cứng, hay database server bị down.
* Lỗi fatal. Những lỗi ảnh hưởng tới toàn bộ hệ thống. Ví dụ hết bộ nhớ cấp phát.
* Các ngoại lệ có thể được phân làm ba lớp:
* 1.1. Ngoại lệ có thể được xử lý nếu người dùng thực hiện lại thao tác. Ví dụ như thay đổi giá trị nhập vào form và submit lại. Những ngoại lệ này là ngoại lệ về mặt nghiệp vụ, hoặc những ngoại lệ ngẫu nhiên gặp khi hệ thống đang hoạt động bình thường. Chẳng hạn người dùng submit vào lúc có quá nhiều requent, hệ thống không xử lý được, nếu thử lại thì khả năng sẽ hết. Những ngoại lệ này được điều khiển bởi mã của chương trình.
* 1.2. Ngoại lệ không thể được xử lý đơn giản bằng cách thực hiện lại thao tác. Những ngoại lệ này có thể là lỗi hệ thống hoặc những lỗi fatal. Những ngoại lệ này do framework điều khiển.
* 1.3. Những ngoại lệ sinh ra do request không hợp lệ. Chẳng hạn request tới uri không có controller mapping nào. Những ngoại lệ này do framework điều khiển.
* Nhà phát triển phải để tâm đến các ngoại lệ trong phân lớp 1.1. Trong khi đó, người có vai trò kiến trúc sư phải để ý đến phân lớp số 1.2 và 1.3.

Có bốn phương thức để xử lý ngoại lệ:

* 2.1. Sử dụng try-catch. Phương pháp này thường để xử lý các ngoại lệ tại request (Controller).
* 2.2. Sử dụng @ExceptionHandler để ủy quyền cho front controller xử lý ngoại lệ.
* 2.3. Sử dụng HandlerExceptionResolver. Phương pháp này được dùng để xử lý ngoại lệ tại servlet.
* 2.4. Sử dụng trang báo lỗi. Phương pháp này được dùng để bắt các fatal error hay các ngoại lệ sinh ra ngoài phạm vi của Spring.

Câu 47:

| **Cookie** | **Session session là gì session là gì** |
| --- | --- |
| **Cookie** được lưu trữ trên trình duyệt của người dùng. | **Session** không được lưu trữ trên trình duyệt. |
| Dữ liệu **cookie** được lưu trữ ở phía client. | Dữ liệu **session** được lưu trữ ở phía server. |
| Dữ liệu **cookie** dễ dàng sửa đổi hoặc đánh cắp khi chúng được lưu trữ ở phía client. | Dữ liệu **session** không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. |
| Dữ liệu **cookie** có sẵn trong trình duyệt đến khi expired. | Sau khi đóng trình duyệt sẽ hết phiên làm việc (session) |

, **[Session](https://topdev.vn/blog/session-la-gi-cookie-la-gi/)** là một phiên làm việc là một khái niệm phổ biến được dùng trong lập trình web có kết nối với database

Một **session** bắt đầu khi client gửi request đến server, nó tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng web và chỉ kết thúc khi hết thời gian timeout hoặc khi bạn đóng ứng dụng. Giá trị của session sẽ được lưu trong một file trên server.

## **Cookie là gì?**

Giống khái niệm session là gì, khái niệm **cookie** cũng được dùng để lưu những thông tin tạm thời. Nhưng tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của bạn khi bạn truy cập vào ứng dụng.

## **Cách sử dụng [Cookie](https://topdev.vn/blog/?s=Cookie)**

**Cookie** thường được tạo ra khi người dùng truy cập một website, **cookie**sẽ ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tuỳ chọn do người dùng lựa chọn đi kèm. Các thông tin này được lưu trong máy tính để nhận biết người dùng khi truy cập vào một trang web.

Câu 50:

Một web service là cách giao tiếp giữa hai hoặc nhiều ứng dụng được phát triển cho các nền tảng khác nhau. Các service này độc lập với các trình duyệt và hệ điều hành qua đó dễ dàng giao tiếp và nâng cao hiệu suất để phù hợp với nhiều người dùng. Service này có thể như một hàm, một bộ sưu tập các tiêu chuẩn hoặc các giao thức được triển khai trên server. Nó là một giao tiếp giữa một client và server hoặc thông tin liên lạc giữa hai thiết bị thông qua kết nối mạng.

Service được phát triển trong Java và được public lên Internet. Hiện nay service có thể được sử dụng bởi bất kỳ ứng dụng nào trên Java, và cả các ứng dụng .NET, Linux, Python. Giao tiếp này được thực hiện thông qua một tập hợp XML-based message qua giao thức HTTP.

Câu 52: REST là viết tắt của Representational State Transfer (chuyển trạng thái đại diện) là một kiểu kiến trúc lập trình. nó định nghĩa các quy tắc để thiết kết các web service chú trọng vào tài nguyên hệ thống Trong kiến trúc REST mọi thứ đều được coi là tài nguyên. Nó có thể ở dạng hiển thị, chẳng hạn như XML, JSON hoặc HTML. Trong RESTful web services, trạng thái của tài nguyên quan trọng hơn hành động được thực hiện đối với tài nguyên.

**RESTful** API **là** một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource.

Jquery : JQuery là một thư viện JavaScript gọn nhẹ (lightweight), "viết ít hơn, làm nhiều hơn" (write less, do more). Mục đích của jQuery là giúp sử dụng JavaScript dễ dàng hơn trên trang web của bạn. JQuery thực hiện rất nhiều công việc phổ biến mà đòi hỏi phải viết nhiều dòng mã JavaScript, và đưa chúng vào các phương thức mà bạn có thể gọi chỉ với một dòng mã.

JQuery cũng đơn giản hoá rất nhiều những thứ phức tạp của JavaScript, như gọi AJAX và thao tác DOM .

AJAX là viết tắt của JavaScript và XML không đồng bộ (Asynchronous JavaScript and XML).  
AJAX là một kỹ thuật mới để tạo ra các ứng dụng web tốt hơn, nhanh hơn và tương tác hơn  
với sự trợ giúp của XML, HTML, CSS và Java Script.

1. Từ trình duyệt của client, ta có một sự kiện để gọi ajax. Khi đó javascript sẽ tạo nên một đối tượng XMLHttpRequest. Và đối tượng này sẽ được gửi một request đến server  
   ví dụ: Khi người dùng click vào ô input và chọn skill tìm việc làm của người dùng. Khi đó ta sẽ lấy thông tin đó và gửi đến server và cần trả về các việc làm tương ứng phù hợp với người dùng.
2. Khi server nhận được HttpRequest từ đó sẽ xử lý request và trả về response cho web.  
   Server xử lí lấy ra các [việc làm javascript](https://topdev.vn/viec-lam-it/javascript--kt22) chẳng hạn thuộc skill mà người dùng cần và trả về dữ liệu của các việc làm.
3. Sau khi nhận được một response từ server, Javascript sẽ xử lý và cập nhật vào trang web cho chúng ta.

Header request parameters là các thông số kèm theo khi gửi yêu cầu lên server

Header response parameters là các thông số kèm theo khi server gửi kết quả về cho client.

I18n - Quốc tế hóa - Cung cấp một số loại khung để bạn có thể dễ dàng trao đổi các chuỗi, đồ họa, âm thanh và các tài nguyên khác và thường xử lý các quy ước giao tiếp khác nhau.

L10n - Bản địa hóa - Quá trình tạo ra một chuỗi các chuỗi, đồ họa, âm thanh, vv để bạn có thể nhắm mục tiêu một quốc tịch, ngôn ngữ hoặc khu vực cụ thể.

**Trong** Spring **MVC** hỗ trợ 1 tính năng khá giống với Filter **trong** JSP-Server đó **là Interceptor**. **Interceptor** thực hiện lọc các request được xử lý bởi controller. Mỗi request gửi đến 1 URL được xử lý bởi controller có thể đi qua không, một hoặc nhiều **Interceptor**.

Security : Về cốt lõi, Spring Security thực sự chỉ là một loạt các bộ lọc servlet giúp bạn thêm xác thực và ủy quyền vào ứng dụng web của mình.

Nó cũng tích hợp tốt với các frameworks như Spring Web MVC (hoặc Spring Boot), cũng như với các tiêu chuẩn như OAuth2 hoặc SAML. Và nó tự động tạo các trang đăng nhập / đăng xuất và bảo vệ chống lại các hành vi khai thác phổ biến như CSRF.

**1. Xác thực (Authentication)**

Trước hết, nếu bạn đang chạy một ứng dụng (web) điển hình, bạn cần người dùng của mình xác thực. Điều đó có nghĩa là ứng dụng của bạn cần xác minh xem người dùng có phải là người anh ta tuyên bố hay không, thường được thực hiện bằng kiểm tra tên người dùng và mật khẩu.

Người dùng: "Tôi là tổng thống Hoa Kỳ. Tên người dùng của tôi là: potus!"

Ứng dụng web của bạn: "Chắc chắn rồi, mật khẩu của bạn lúc đó là gì, thưa Tổng thống?"

Người dùng: "Mật khẩu của tôi là: th3don4ld".

Ứng dụng web của bạn: "Đúng. Chào mừng, thưa ngài!"

**2. Ủy quyền (Authorization)**

Trong các ứng dụng đơn giản hơn, xác thực có thể là đủ: Ngay sau khi người dùng xác thực, họ có thể truy cập mọi phần của ứng dụng.

Nhưng hầu hết các ứng dụng đều có khái niệm về quyền (hoặc vai trò). Hãy tưởng tượng: khách hàng có quyền truy cập vào giao diện công khai của webshop của bạn và quản trị viên có quyền truy cập vào khu vực quản trị riêng biệt.

Cả hai loại người dùng đều cần đăng nhập, nhưng thực tế xác thực đơn thuần không nói lên điều gì về những gì họ được phép làm trong hệ thống của bạn. Do đó, bạn cũng cần kiểm tra các quyền của người dùng đã xác thực, tức là bạn cần cấp quyền cho người dùng đó.

Người dùng: "Hãy để tôi chơi với cái vali hạt nhân đó…."

Ứng dụng web của bạn: "Một giây, tôi cần kiểm tra các quyền của ông trước ... .. vâng thưa Tổng thống, ông có đủ điều kiện phù hợp. Hãy tận hưởng."

Người dùng: "Nút đỏ đó lại là gì ... ??"

**3. Bộ lọc Servlet (Servlet Filters)**

Cuối cùng nhưng không kém phần quan trọng, chúng ta hãy xem xét Bộ lọc Servlet. Họ phải làm gì với xác thực và ủy quyền?

Tại sao sử dụng Bộ lọc Servlet?

CORS: Cross-origin resource sharing (viết tắt là CORS) là một tiêu chuẩn để truy cập tài nguyên web trên các tên miền khác nhau. CORS cho phép các web scripts tương tác cởi mở hơn với nội dung bên ngoài tên miền gốc, dẫn đến sự tích hợp tốt hơn giữa các dịch vụ web.

Trong trường hợp đơn giản nhất, phía client sẽ tạo request GET, POST, PUT, DELETE để yêu cầu server thực hiện một việc gì đó. Những request này sẽ được đính kèm một header tên là Origin để chỉ định origin của client code (giá trị của header này chính là domain của trang web).

Server sẽ xem xét Origin để biết được nguồn này có phải là nguồn hợp lệ hay không. Nếu hợp lệ, server sẽ trả về response kèm với header Access-Control-Allow-Origin. Header này sẽ cho biết xem client có phải là nguồn hợp lệ để browser tiếp tục thực hiện quá trình request.

Trong trường hợp thông thường, Access-Control-Allow-Origin sẽ có giá trị giống như Origin, một số trường hợp giá trị của Access-Control-Allow-Origin sẽ nhìn giống giống như Regex hay chỉ đơn giản là \*, tuy nhiên thì cách dùng \* thường được coi là không an toàn vì API của bạn sẽ được public hoàn toàn và ai cũng có thể truy cập được.

**CSRF** hay còn gọi là kỹ thuật tấn công “**Cross-site Request Forgery**“, nghĩa là kỹ **thuật tấn công giả mạo** chính chủ thể của nó. **CSRF** nói đến việc tấn công vào chứng thực request trên web thông qua việc sử dụng Cookies, nơi mà các hacker có khả năng sử dụng thủ thuật để tạo request mà bạn không hề biết. Vì vậy, một CSRF là hacker lạm dụng sự tin tưởng của một ứng dụng web trên trình duyệt của nạn nhân.

CSRF là một kiểu tấn công gây sự nhầm lẫn tăng tính xác thực và cấp quyền của nạn nhân khi gửi một request giả mạo đến máy chủ. Vì thế một lỗ hổng CSRF ảnh hưởng đến các quyền của người dùng ví dụ như quản trị viên, kết quả là chúng truy cập được đầy đủ quyền.

**CSRF** là một kiểu tấn công gây sự nhầm lẫn tăng tính xác thực và cấp quyền của nạn nhân khi gửi một request giả mạo đến máy chủ. Vì thế một lỗ hổng CSRF ảnh hưởng đến các quyền của người dùng ví dụ như quản trị viên, kết quả là chúng truy cập được đầy đủ quyền.

JWT là viết tắt của từ JSON Web Token là một tiêu chuẩn mở (([RFC 7519](https://tools.ietf.org/html/rfc7519)) định nghĩa một cách thức truyền tin an toàn giữa các thành viên bằng 1 đối tượng JSON. Thông tin này có thể được xác thực và đánh dấu là tin cậy dựa vào chữ ký của nó và phần chữ ký này của JWT sẽ dược mã hóa với thuật toán **HMAC**hoặc sử dụng cặp khóa public/private key của thuật toán **RSA.**

Spring Boot 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). Spring Boot được sử dụng để tạo ra một ứng dụng độc lập dựa trên Spring mà có thể chạy ngay với rất ít cấu hình cần thêm. Spring Boot không cần đến bất kỳ cấu hình xml nào, mà nó sử dụng quy ước về mô hình thiết kế phần mềm theo cấu hình, có nghĩa là nó giảm tải công việc cho các lập trình viên.

Tomcat server được cấu hình nhúng sẵn trong một ứng dụng Spring Boot. • Từ các thư viện dependencies đã khái báo, Spring Boọt tự động cấu hình các thành phần cần thiết cho ứng dụng

Đặc điểm auto-configuration của Spring Boot sẽ tự động cấu hình dự án dựa trên các dependencies đã được thêm vào Gradle/Maven. • Khởi động auto-configuration thông qua antonation @EnableAutoconfiguration hoặc @SpringBootApplication

| **Spring Boot** | **Spring MVC** |
| --- | --- |
| **Spring Boot** là một module của Spring để đóng gói ứng dụng dựa trên Spring với các giá trị mặc định hợp lý. | **Spring MVC** model view controller-based web framework dựa trên Spring. |
| Nó cung cấp các cấu hình mặc định để xây dựng **Spring-powered framework**. | Nó cung cấp các tính năng sẵn sàng sử dụng để xây dựng một ứng dụng web. |
| Không có yêu cầu đối với các file mô tả khi triển khai. | Cần có nhiều file mô tả triển khai. |
| Nó tránh mã viết sẵn và kết hợp các dependencies lại với nhau trong một đơn vị duy nhất. | Nó chỉ định từng dependency riêng biệt. |
| Nó làm giảm thời gian phát triển và tăng năng suất. | Cần nhiều thời gian hơn để đạt được điều tương tự. |