**TESTING**

unit testing là một phương pháp kiểm thử phần mềm mà mỗi đơn vị mã nguồn,

**JUnit** là một framework testing cho các ứng dụng Java. Nó cung cấp các annotation và assetion để giúp các mình tạo và chạy test tự động

**Mockito** là một thư viện dựa trên Java được sử dụng để tạo và định cấu hình các đối tượng mô phỏng để mô phỏng hành vi của các thành phần đang được thử nghiệm. cho mục đích testing.

Để mô phỏng một lớp bằng Mockito, trước tiên bạn sẽ sử dụng phương thức tĩnh mock() để tạo một đối tượng mô phỏng của lớp đó. Sau đó, bạn có thể sử dụng các phương thức như when() và thenReturn() để định cấu hình hành vi của đối tượng mô phỏng.

Có một số lý do tại sao bạn có thể chọn sử dụng Mockito trong các bài kiểm tra Java của mình:

- Cải thiện khả năng kiểm tra: Bằng cách mo phong cac doi tuong phức tạp và khó quản lý bằng các đối tượng giả đơn giản, các bài kiểm tra của bạn trở nên dễ đọc và dễ bảo trì hơn,

Tiết kiệm thời gian và tài nguyên : Bằng cách sử dụng Mocks, bạn có thể tránh khởi động các đối tượng thực, có thể mất một lượng thời gian và tài nguyên đáng kể, như cơ sở dữ liệu, lệnh gọi bên ngoài.

Tăng tốc độ thử nghiệm của bạn: Bằng cách tách biệt mã đang thử nghiệm khỏi các phần dependencies cua no, bạn có thể chạy thử nghiệm đơn vị nhanh hơn và giúp quá trình phát triển hiệu quả hơn.

*Cách ly các phần phụ thuộc: bạn có thể tách biệt code mà bạn đang kiểm tra khỏi các dependencies, kiểm tra code một cách độc lập và dễ dàng xác định các vấn đề hơn.*

**BACKEND**

Khi chạy chương trình Java, JVM sẽ yêu cầu hệ điều hành (Windows, Linux, Mac,..) cấp phát một không gian bộ nhớ trong RAM. JVM sẽ phân chia không gian bộ nhớ này được cấp phát thành các vùng nhớ khác nhau để lưu những dữ liệu cần thiết cho việc chạy chương trình.

**IOC container**

ioC là framework dùng để quản lí và cấu hình đối tượng trong java. ioc này sẽ chịu trách nhiệm tạo và đưa dependencies vào đối tượng cần thiết. giúp tách rời các thành phần khác nhau của ứng dụng, dễ quản lí và duy trì code

**cách hoat dong:** bằng cách sử Dependency injention để quản lý việc tạo và cấu hình các đối tượng trong ứng dụng. Khi cần một đối tượng, bộ chứa IoC sẽ tạo đối tượng và đưa bất kỳ dependency cần thiết nào vào đối tượng đó.

Về cơ bản có hai loại IOC Container trong Spring: ApplicationContext, BeanFactory: BeanFactory giống như một lớp nhà máy chứa một tập hợp các hạt đậu. Nó khởi tạo bean bất cứ khi nào co yêu cầu. ApplicationContext: Giao diện ApplicationContext được xây dựng trên giao diện BeanFactory.

**Dependency Injection là gì**

là software design pattern cho phép cung cấp các dependency bên ngoài cho một đối tượng hoặc lớp.

**Cong dung** Điều này giúp tách đối tượng hoặc lớp khỏi các dependency của nó, giúp dễ dàng thay đổi hoặc hoán đổi các dependency và giảm độ phức tạp tổng thể của ứng dụng.

**A design pattern in Java**  là một giải pháp có thể tái sử dụng cho một vấn đề thiết kế phần mềm phổ biến.

**SCRUM là framework quan li va phat trien du an. (search more)**

**Minh bạch, Thanh tra, thích nghi,**

****Transactions** được sử dụng để đảm bảo rằng các hoạt động của cơ sở dữ liệu là nguyên tử, nhất quán, biệt lập và lâu bền.**

Nguyên tử có nghĩa là một giao dịch, tất cả các hoạt động trong giao dịch đều được hoàn thành thành công hoặc không có hoạt động nào được hoàn thành. Tính nhất quán có nghĩa là một giao dịch đưa cơ sở dữ liệu từ trạng thái hợp lệ này sang trạng thái hợp lệ khác. Bị cô lập có nghĩa là tác động của một giao dịch không hiển thị đối với các giao dịch khác cho đến khi giao dịch được thực hiện. Bền có nghĩa là tác động của giao dịch vẫn tồn tại ngay cả khi hệ thống bị lỗi.

**SQL Injection** là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn [SQL](https://topdev.vn/blog/sql-la-gi/) để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database. **SQL injection** có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

**ALL Tương ứng với tất cả các loại cascade. cascade={DETACH, MERGE, PERSIST, REFRESH, REMOVE}**

**DETACH Nếu đối tượng cha bị detached khỏi persistence context(ngữ cảnh tồn tại) thì các đối tượng tham chiếu tới nó cũng bị detached.**

**MERGE Nếu đối tượng cha được merged vào persistence context, thì các đối tượng tham chiếu tới nó cũng được merged.**

**PERSIST Nếu đối tượng cha được persisted vào persistence context, thì các đối tượng tham chiếu tới nó cũng được persisted.**

**REFRESH Nếu đối tượng cha được refreshed ở persistence context hiện tại, thì các đối tượng tham chiếu tới nó cũng được refreshed.**

**REMOVE Nếu đối tượng cha bị removed khỏi persistence context, thì các đối tượng tham chiếu tới nó cũng được removed.**

Mặc định sẽ không có sửa, update vì nó sẽ ảnh hưởng tới các employee đang tham chiếu tới.

(ON UPDATE/DELETE = NO ACTION, RESTRICT)

2.2: set company\_id của các employee đang tham chiếu tới bằng null (ON UPDATE/ON DELETE = SET NULL)

(Tạm hiểu là công ty bị xóa, thay đổi thì các nhân viên sẽ không thuộc công ty nào cả)

2.3: set company\_id của các employee bằng id của company sau khi update (cho trường hợp update)

. Và xóa tất cả các employee có company\_id tham chiếu tới company bị xóa (ON UPDATE/ON DELETE = CASCADE

There are 5 bean scopes in spring framework.

|  |  |  |
| --- | --- | --- |
| **No.** | **Scope** | **Description** |
| 1) | singleton | The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope. |
| 2) | prototype | The bean instance will be created each time when requested. |
| 3) | request | The bean instance will be created per HTTP request. |
| 4) | session | The bean instance will be created per HTTP session. |
| 5) | globalsession | The bean instance will be created per HTTP global session. It can be used in portlet context only. |

**JWT**

JWT là json web token, là chuẩn json web token giúp tạo ra chuỗi mã hóa chứa dữ liệu để trao đổi thông tin giữa hệ thống khác nhau

(client va server)an toàn đáng tin cậy, cấu trúc gồm 3 phần: header payload, Signature

*Đưa vào hàm authenticate của cái thằng authentication này chính là gì thì chính là một cái*

*interface nó được lưu sẵn của spring nó giúp cho cái việc xác thực jwt sau khi xác thực xong*

*thi mình sẽ lưu phần xác thực đó vào getContext của SecurityContextHolder và từ các*

*thành phần này mình sẽ tạo ra được cái một cái chuỗi jwt dựa trên jwtUtil (class do mình viết)*

**UUID**

UUID thực là một số nhận dạng duy nhất được tạo và biểu diễn bằng một định dạng chuẩn hóa.

UUID hợp lệ được xác định bởi RFC RFC bao gồm năm thuật toán khác nhau

Dựa trên thời gian

Bảo mật DCE

Dựa trên tên (MD5)

Ngẫu nhiên

Dựa trên tên (SHA-1)

**FETCHTYPE**

fetch = FetchType.LAZY tức là khi bạn find, select đối tượng Company từ database

thì nó sẽ không lấy các đối tượng Employee liên quan

fetch = FetchType.EAGER tức là khi bạn find, select đối tượng Company từ database

thì tất cả các đối tượng Employee liên quan sẽ được lấy ra và lưu vào listEmployee

**CI/CD** stands for Continuous Integration and Continuous Deployment

đó là một phương pháp phát triển phần mềm liên quan đến việc tự động xây dựng, thử nghiệm và triển khai các thay đổi mã. Nó giúp giảm lỗi của con người, cải thiện sự hợp tác và tăng tốc độ cung cấp phần mềm.

automated testing, tăng chất lượng mã, phân phối phần mềm nhanh hơn, tăng cường cộng tác, cải thiện khả năng theo dõi và khả năng hiển thị các thay đổi mã.

**, CORS** là một tính năng bảo mật được triển khai bởi các trình duyệt web để ngăn các trang web độc hại thực hiện các yêu cầu trái phép thay cho người dùng.

**CSRF** là một kỹ thuật tấn công mà người tấn công sử dụng một trang web khác để gửi yêu cầu bất hợp pháp đến một trang web khác, sử dụng thông tin đăng nhập của người dùng để thực hiện các hành vi không mong muốn.

Trong Java, CSRF có thể được ngăn chặn bằng cách sử dụng mã xác thực trên các yêu cầu để xác minh rằng yêu cầu đó đã được gửi từ trang web được tin cậy. Các thư viện như Spring Security cung cấp các công cụ hỗ trợ để bảo vệ khỏi CSRF trong Java. Người dùng cũng có thể sử dụng các tiện ích như mã xác thực trên phiên hoặc cookie để tránh CSRF.

|  |  |  |
| --- | --- | --- |
| **Tham số** | **HashSet** | **HashMap** |
| Thực thi | Giao diện set | Giao diện Map |
| Lưu trữ | Các đối tượng | Các cặp khóa - giá trị |
| Nhân bản giá trị | HashSet không cho phép các giá trị trùng lặp | HashMap cho phép các khóa trùng lặp nhưng không cho phép các giá trị trùng lặp |
| Giá trị Null | Cho phép một giá trị Null | Cho phép một khóa null nhưng nhiều giá trị null |
| Phương thức chèn phần tử | Chúng ta sử dụng phương thức add() để chèn các phần tử trong HashSet | Chúng ta sử dụng phương thức put  () để chèn các phần từ trong HashMap |
| Số đối tượng | Chúng ta chỉ có thể tạo một đối tượng trong quá trình thêm các phần tử | Chúng ta tạo hai đối tượng trong quá trình thêm các đối tượng |
| Hiệu suất | Chậm hơn | Nhanh hơn |
| Cách dùng | Sử dụng HashSet để duy trì tính duy nhất của dữ liệu | Sử dụng HashMap là khi tính duy nhất của dữ liệu |

|  |  |  |
| --- | --- | --- |
| ArrayList |  | Linkedlist |
| 1) | ArrayList sử dụng một mảng động. | LinkedList sử dụng danh sách liên kết doubly. |
| 2) | ArrayList không hiệu quả với thao tác vì cần nhiều chuyển đổi. | LinkedList là hiệu quả cho thao tác. |
| 3) | ArrayList là tốt hơn để lưu trữ và lấy dữ liệu. | LinkedList là tốt hơn để thao tác dữ liệu. |

|  |  |  |
| --- | --- | --- |
| No. | HashMap | Hashtable |
| 1) | HashMap là KHÔNG synchronized. | Hashtable là synchronized. |
| 2) | HashMap có thể chứa một khóa null và nhiều giá trị null. | Hashtable không thể chứa bất kỳ khóa null hoặc giá trị null. |

|  |  |
| --- | --- |
| Array | ArrayList |
| 1) Kích thước **cố định**. | Kích thước có thể **thay đổi được**. |
| 2) Có thể lưu trữ dữ liệu kiểu **nguyên thủy** và **đối tượng**. | Chỉ có thể lưu trữ dữ liệu kiểu **đối tượng**. Kể từ Java 5, kiểu nguyên thủy được tự động chuyển đổi trong các đối tượng được gọi là **auto-boxing**. |
| 3) Tốc độ lưu trữ và thao tác **nhanh hơn**. | Tốc độ lưu trữ vào thao tác **chậm hơn**. |
| 4) Chỉ có thuộc tính **length**. | Có nhiều phương thức để thao tác với dữ liệu. |

|  |  |
| --- | --- |
| **HashMap** | **HashTable** |
| HashMap is not synchronized thereby making it better for non-threaded applications. | HashTable is synchronized and hence it is suitable for threaded applications. |
| Allows only one null key but any number of null in the values. | This does not allow null in both keys or values. |
| Supports order of insertion by making use of its subclass LinkedHashMap. | Order of insertion is not guaranteed in HashTable. |

**RuntimeException** là lớp cơ sở của tất cả các lớp unchecked exception. Đó cũng là dấu hiệu để nhận biết đâu là

checked exception và đâu là unchecked exception.

Điểm khác biệt giữa các lớp checked và unchecked expcetion chính là thời điểm xác định được expcetion có thể xảy ra.

@ControllerAdvice là một Annotation gắn trên Class. Có tác dụng xen vào quá trình xử lý của các @RestController.thường được kết hợp với @ExceptionHandler để cắt ngang quá trình xử lý của Controller, và xử lý các ngoại lệ xảy ra.

Hiểu đơn giản là Controller đang hoạt động bình thường, chẳng may có một Exception được ném ra, thì thay vì báo lỗi hệ thống,thì nó sẽ được thằng @RestControllerAdvice và @ExceptionHandler đón lấy và xử lý. Sau đó trả về cho người dùng thông tin hữu ích hơn.

cách thêm annotation @ExceptionHandler trước method đó để chỉ dẫn rằng method đó sẽ xử lý cho exception nào

**@RestController** **là một @Controller chuyên dùng để tạo REST API**. Còn @Controller khai báo bean component để hứng và xử lý request rồi trả về response nói chung.

**@RequestBody và @ResponseBody**   **Dùng để ánh xạ dữ liệu truyền từ client lên server và ngược lại**.

@RequestParam annotation được **sử dụng để trích xuất dữ liệu từ query parameters, form parameters, và kể cả các tập tin từ request**.

**FE**

**COOKIES**

Cookie là những tập tin một trang web gửi đến máy người dùng và được lưu lại thông

qua trình duyệt khi người dùng truy cập trang web đó. Cookie được dùng với mục đích

phổ biến là để lưu trữ phiên đăng nhập nhằm phục vụ cho mục đích xác thực với website,

duy trì trạng thái đăng nhập.

**local storage** là một feature mới của HTML5 cho phép bạn lưu trữ bất kì info nào bạn muốn trong browser dùng JavaScript. Nó rất là đơn giản.

Thực tế thì local storage chỉ là một object JavaScript bự và cũ kĩ cho phép bạn attach data (hoặc remove cũng được).

**FIxed**

Tương tự như thuộc tính Absolute, Fixed cũng hoạt động dựa vào một thẻ cha làm chuẩn bên ngoài. Chỉ có khác một điều là đối

với Fixed thì khi bạn scrolling thì nó sẽ nằm luôn trên màn hình ngay tại vị trí chuẩn đó và đè lên trên các phần tử bên dưới.

**REF**

REF: chúng ta làm việc với DOM elements bằng cách gọi document.getElementById().

Với ref trong React chúng ta không cần phải làm vậy.

Thuộc tính ref sẽ tham chiếu đến chính xác element cần dùng.

useRef hook là một function trả về một object với thuộc tính current được khởi tạo thông

qua tham số truyền vào. Object được trả về này có thể mutate và sẽ tồn tại xuyên suốt vòng đời của component.

Về cơ bản, useRef giống như một "box", dùng để lưu các giá trị của một element sử dụng thuộc tính .current.

Nếu bạn chuyển một đối tượng ref cho React với <div ref = {myRef} />, React sẽ đặt thuộc tính .current của

nó tới node DOM tương ứng bất cứ khi nào node đó thay đổi.

useEffect Hook như componentDidMount, componentDidUpdate và componentWillUnmount được kết hợp.

useEffect sẽ giúp bạn xử lý các logic trong vòng đời của component và được gọi khi component có sự thay đổi.

useState cho phép chúng ta khai báo local state trong Function Component cách mà trước để chỉ dùng cho Class Component.

định nghĩa tên của state nó có thể là đơn giá trị hoặc object

**PROPS**

props là là 1 từ viết ngắn gọn của properties ,props là 1 đối tượng,

nó lưu trữ các giá trị của các attribute (thuộc tính) của một thẻ (Tag)

cách mà component có thể nhận được các giá trị của thuộc tính truyền vào từ bên ngoài vào, và là cách mà các

component có thể nói chuyện với nhau.

**State**

* Khái niệm: Cũng giống như props , state cũng lưu trữ thông tin về component, nhưng là lưu trữ dữ liệu động của một component.
* State là dữ liệu động , nó cho phép một component theo dõi thông tin thay đổi giữa các kết xuất (render) và làm cho nó có thể tương tác.
* State chỉ có thể được sử dụng ở trong một component sinh ra nó.
* Nếu dự đoán được một component cần quản lý state, thì nó nên được tạo ở trong một class component chứ không phải là một function component.

Props state

|  |  |  |
| --- | --- | --- |
| Có thể nhận giá trị ban đầu từ component cha | Yes | Yes |
| Có thể thay đổi bởi component cha | Yes | No |
| Có thể set giá trị mặc định bên trong component | Yes | Yes |
| Có thể thay đổi bên trong component | No | Yes |
| Có thể set giá trị ban đầu cho các component con | Yes | Yes |
| Có thể thay đổi trong các component con | Yes | No |

**Other Question**

**Để tránh trường hợp 2 người cùng lúc update một record trong cùng một khoảng thời gian, có một số giải pháp mà bạn có thể thử:**

1. Sử dụng khóa dữ liệu (database lock): Đây là một tính năng của hầu hết các cơ sở dữ liệu cho phép bạn khóa một record trong bảng để tránh trường hợp hai người truy cập vào cùng lúc. Tuy nhiên, khóa dữ liệu có thể gây chậm trễ cho hệ thống nếu sử dụng không đúng cách.
2. Sử dụng optimistic locking: Trong cách này, mỗi record sẽ có một trường "version" được tăng lên mỗi lần update. Nếu hai người cùng lúc update cùng một record, chỉ có người có version lớn hơn mới có thể update thành công.
3. Sử dụng pessimistic locking: Trong cách này, khi một người muốn update một record, hệ thống sẽ khóa record đó cho đến khi người đó xong việc update rồi mới mở khóa. Điều này có thể gây chậm trễ cho hệ thống nếu có quá nhiều yêu cầu update đồng thời.

**Triển khai dự án lên môi trường thật: Có nhiều cách để triển khai dự án lên môi trường thật, bao gồm:**

* Sử dụng công cụ quản lý container như Docker để đóng gói và triển khai dự án trên môi trường thật.
* Triển khai trên máy chủ web như Tomcat hoặc Jetty bằng cách đặt file war trong thư mục webapps của Tomcat hoặc Jetty.
* Triển khai trên môi trường PaaS (Platform as a Service) như Heroku hoặc OpenShift bằng cách sử dụng công cụ để đẩy dự án lên môi trường PaaS.

**DOCKER** công cụ được thiết kế để giúp tạo, triển khai và chạy các ứng dụng dễ dàng hơn bằng cách sử dụng cac container. container cho phép mình đóng gói một ứng dụng với tất cả các phần mà nó cần, chẳng hạn như thư viện và các thành phần phụ thuộc khác, đồng thời vận chuyển tất cả ứng dụng dưới dạng một gói. Bằng cách đó, Docker cho phép mình chạy các ứng dụng trong mọi môi trường, trên mọi thiết bị mà không phải lo lắng về sự khác biệt trong thiết lập hoặc phụ thuộc.

**Old Project**

**JavaFX** is a software platform for creating and delivering desktop applications,

một số tính năng chính của JavaFX bao gồm:

Biểu đồ cảnh: JavaFX sử dụng biểu đồ cảnh để thể hiện các thành phần trực quan của ứng dụng. Biểu đồ cảnh là một cây phân cấp gồm các nút đại diện cho các hình dạng, văn bản và hình ảnh tạo nên giao diện người dùng.

Tạo kiểu CSS: JavaFX cho phép tạo kiểu cho giao diện người dùng bằng cách sử dụng CSS (Cascading Style Sheets). Điều này giúp dễ dàng thay đổi giao diện của ứng dụng và tách bố cục khỏi logic nghiệp vụ.

FXML: JavaFX bao gồm một ngôn ngữ khai báo có tên là FXML, ngôn ngữ này cho phép xác định giao diện người dùng trong một tệp riêng biệt với logic nghiệp vụ. Điều này có thể làm cho việc duy trì và sửa đổi bố cục của một ứng dụng trở nên dễ dàng hơn.

Đồ họa: JavaFX bao gồm một công cụ đồ họa mạnh mẽ có thể được sử dụng để tạo các hình ảnh trực quan tương tác, phong phú. Nó hỗ trợ đồ họa 2D và 3D, cũng như tăng tốc phần cứng trên các nền tảng được hỗ trợ.

**JDBC** là một API Java cho phép các nhà phát triển kết nối và thao tác cơ sở dữ liệu từ các ứng dụng Java.

Driver Manager: : Trình quản lý trình điều khiển chịu trách nhiệm quản lý các trình điều khiển JDBC có sẵn cho ứng dụng. Nó chịu trách nhiệm thiết lập các kết nối với cơ sở dữ liệu và định tuyến các yêu cầu tới trình điều khiển thích hợp.

Driver: Trình điều khiển JDBC chịu trách nhiệm dịch các lệnh gọi JDBC thành các lệnh gọi API gốc được sử dụng bởi cơ sở dữ liệu. Có một số loại trình điều khiển có sẵn, bao gồm Loại 1 (cầu nối JDBC-ODBC), Loại 2 (trình điều khiển API gốc/một phần Java) và Loại 4 (trình điều khiển Java thuần túy).

Connection: Connection nối đại diện cho một phiên làm việc với cơ sở dữ liệu. Nó được sử dụng để tạo câu lệnh, gửi truy vấn và truy xuất kết quả.

Câu lệnh: Câu lệnh được sử dụng để thực thi câu lệnh SQL và truy xuất kết quả. Có ba loại câu lệnh: Statement, PreparedStatement và Callable Statement.

Bộ kết quả: Bộ kết quả biểu thị kết quả của một truy vấn SQL. Nó có thể được sử dụng để lặp qua các hàng của kết quả và truy xuất dữ liệu cho mỗi cột.

**Hibernate** là framework ánh xạ quan hệ đối tượng (ORM) mã nguồn mở cho ngôn ngữ Java. Nó được sử dụng để ánh xạ các đối tượng Java tới các bảng cơ sở dữ liệu và ngược lại, cho phép mình làm việc với dữ liệu được lưu trữ trong cơ sở dữ liệu bằng các đối tượng Java. CONG^ DUNG: Tự động tạo SQ , Độc lập với CSDL,

2. Jps là gì? Ưu điểm của jps so với asp?

**JSP là viết tắt của JavaServer Pages**là một công nghệ để phát triển các trang web động. JSP giúp các nhà phát triển chèn java code vào các trang HTML bằng cách sử dụng các thẻ JSP đặc biệt.

4. Hãy cho biết thế nào là mô hình MVC?

**MVC** là viết tắt của cụm từ “**Model-View-Controller.** MVC là một **mẫu kiến trúc phần mềm** để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau như tên gọi: Model (dữ liệu), View (giao diện) và Controller (bộ điều khiển).

Đơn giản hơn, là mô hình này được chia thành 3 phần trong soure code. Và mỗi phần đảm nhận vai trò và nhiệm vụ riêng biệt nhau và độc lập.

https://viettuts.vn/interview/list-cau-hoi-phong-van-java-collection