Java Full Stack Developer Interview Questions and Answers

Introduction

The Java Full Stack Developer is one of the first steps that we took in combining Java and Full Stack together. Combining these two courses will give candidates an expanded knowledge of both Java and Full Stack, which will make them understand the course better together. Now, with the integration of Java and Full Stack together, the combined demand for a Java Full Stack Developer has never been higher, and these interview questions and answers will increase your chances of getting hired easily. Click here to learn more about **our** **Java Full Stack Developer Syllabus**.

List of Java Full Stack Developer Interview Questions for Freshers

1. What is Java Full Stack?

2. What does a Java Full-Stack developer do?
3. Identify the fundamental components of a typical web application.
4. Enumerate the advantages of utilizing Spring Boot for Java web development.
5. Explain the MVC architectural pattern and its implementation in Java web applications.
6. Define RESTful web services and their implementation in Java.
7. Elaborate on dependency injection and its application in the Spring Framework.
8. What techniques contribute to securing a Java web application?
9. Enumerate popular front-end frameworks/libraries frequently used alongside Java back-end.
10. Explain the concept of connection leak in Java Full Stack Development?

Join our **Java Full Stack Developer Online Training** to learn more about Java Full Stack.

Java Full Stack Developer Interview Questions and Answers for Freshers

1. What is Java Full Stack?

Java Full Stack commonly denotes a software development methodology wherein professionals engage in both front-end and back-end aspects of web applications utilizing Java technologies. This encompasses mastery across diverse facets of web development, encompassing server-side programming, client-side programming, and adept handling of databases.

2. What does a Java Full-Stack developer do?

The following are the responsibilities of a Java Full-Stack developer:

- Crafting user interfaces and interactions utilizing technologies such as HTML, CSS, and JavaScript, alongside frameworks like Angular, React, or Vue.js for the front end.
- Developing server-side operations and managing data interactions through Java-based frameworks like Spring Boot, Hibernate, or Apache Struts for the back-end.
- Designing and maintaining databases for easy data storage and retrieval, involving relational databases like MySQL, PostgreSQL, or Oracle, as well as NoSQL databases like MongoDB or Apache Cassandra.

3. Identify the fundamental components of a typical web application.

A typical web application comprises a front-end (client-side) and a back-end (server-side). The front end encompasses HTML, CSS, and JavaScript, while the back end typically involves a server-side language like Java, a database management system (DBMS), and server software.

4. Enumerate the advantages of utilizing Spring Boot for Java web development.

Spring Boot streamlines the process of building Java web applications by providing pre-configured solutions for common tasks such as dependency management and auto-configuration. It embraces convention over configuration, reducing boilerplate code and accelerating development. Additionally, it seamlessly integrates with other Spring projects and libraries.

5. Explain the MVC architectural pattern and its implementation in Java web applications.

- MVC stands for Model-View-Controller. In this pattern, the Model handles data and business logic, the View manages the presentation layer (UI), and the Controller serves as an intermediary that processes user input, updates the model, and manipulates the view.
- In Java web applications, frameworks like Spring MVC or JavaServer Faces (JSF) are commonly employed to implement the MVC pattern.

6. Define RESTful web services and their implementation in Java.

RESTful web services adhere to the principles of Representational State Transfer (REST), utilizing standard HTTP methods (GET, POST, PUT, DELETE) for CRUD operations (Create, Read, Update, Delete) on resources. In Java, RESTful web services can be realized through frameworks like Spring Boot with Spring MVC or JAX-RS (Java API for RESTful Web Services).

7. Elaborate on dependency injection and its application in the Spring Framework.

- Dependency injection is a design pattern aimed at achieving loose coupling between classes by injecting dependencies (objects) into a class rather than having the class create its dependencies.
- In the Spring Framework, dependency injection is facilitated through inversion of control (IoC) containers, such as the ApplicationContext. Spring manages the creation and maintenance of objects (beans) and their dependencies, facilitating easier testing, maintenance, and scalability.

8. What techniques contribute to securing a Java web application?

Several techniques contribute to securing a Java web application, including:

- Establishing authentication and authorization using frameworks like Spring Security.
- Adoption of HTTPS to encrypt data transmitted between the client and server.
- Validation of user input to prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS).
- Utilization of session management techniques to prevent session hijacking and fixation.
- Regular updates of libraries and frameworks to mitigate known security vulnerabilities.

9. Enumerate popular front-end frameworks/libraries frequently used alongside Java back-end.

Popular front-end frameworks/libraries utilized in conjunction with Java back-end include:

- Angular
- React
- Vue.js
- jQuery
- Bootstrap

10. Explain the concept of connection leak in Java Full Stack Development?

In Java Full Stack Development, a connection leak occurs when a database connection isn't closed correctly after it's been used. This results in a pile-up of open connections, which can overwhelm the available connections and lead to performance issues or system crashes.

Learn more with our Java Full Stack Developer Tutorial for Beginners.

List of Java Full Stack Developer Interview Questions for Experienced

1. Explain the concept of Long Polling.
2. Explain the difference between the concepts of GET and POST in Java Full Stack Development.
3. Explain the ways to fix a connection leak in Java Full Stack Development.
4. Explain SOLID principles in Java Full Stack Development.
5. What is an Actuator in Java Full Stack Development?

Java Full Stack Developer Technical Interview Questions and Answers for Experienced

1. Explain the concept of Long Polling.

Long polling is a method for real-time communication between a web browser and a server. Here's how it works:

- **Client Sends Request:** The browser sends a request to the server, usually using AJAX.

- **Server Holds Response:** Instead of immediately replying, the server keeps the request open.
- **Server Waits for Data or Event:** The server waits for new information or an event that the client wants.
- **Server Responds When Ready:** Once there's new data, the server sends it back to the client.
- **Client Processes Response and Sends New Request:** The client handles the data and sends another request for updates.
- **Repeat Process:** This cycle continues, ensuring the client gets timely updates.

2. Explain the difference between the concepts of GET and POST in Java Full Stack Development.

GET Method	POST Method
Used to fetch data from the server.	Used to send data to the server for processing or storage.
Parameters are included in the URL as part of the query string.	Parameters are sent within the request body, not visible in the URL, making it more secure than GET.
Limited data can be sent because the parameters are visible in the URL, which makes it less secure.	No data size limit, suitable for sending large amounts of data.
Cached by the browser, making it good for requests that can be repeated without causing changes.	Not cached by the browser, making it suitable for requests that could cause changes on the server.
Typically used for safe and repeatable operations like retrieving data from a database or getting a webpage.	Generally used for operations that change the server state, like submitting a form, uploading files, or updating a database.

3. Explain the ways to fix a connection leak in Java Full Stack Development.

Follow the steps below to fix a connection leak:

- **Ensure Proper Resource Handling:** Always close database connections in a finally block to guarantee they're properly closed, even during exceptions. This ensures connections are returned to the pool after use.
- **Use Connection Pooling Libraries:** Employ connection pooling libraries like HikariCP, Apache DBCP, or c3p0. These tools manage connections efficiently, handling pooling, recycling, and closure automatically to reduce leak risks.
- **Implement Try-with-Resources:** Utilize try-with-resources statements in Java (introduced from Java 7 onwards). This feature allows for automatic resource management, closing connections when they're no longer required.
- **Perform Code Reviews and Testing:** Regularly review code and conduct testing to pinpoint and resolve potential connection leaks. Pay special attention to sections where database connections are established, ensuring they're correctly closed in all scenarios.
- **Monitor Connection Pool Usage:** Keep an eye on connection pool usage within your application to detect any unusual patterns or leaks. Monitoring helps identify and fix leaks promptly, minimizing their impact on performance and stability.

Explore our Java full-stack developer project ideas to learn more.

4. Explain SOLID principles in Java Full Stack Development.

In Java Full Stack Development, the SOLID principles are a set of five design rules introduced by Robert C. Martin, also known as Uncle Bob. They're meant to make software easier to understand, more adaptable, and simpler to maintain.

Here's a quick overview of each SOLID principle:

- **Single Responsibility Principle (SRP):** Each class should focus on doing just one thing. This means it should have only one job or responsibility, and all its methods should be related to that job.
- **Open/Closed Principle (OCP):** Classes and modules should be designed so they can be extended without changing their existing code. You can add new features without modifying the existing codebase.

- **Liskov Substitution Principle (LSP):** Subclasses should be able to replace their parent classes without changing how the program works. This ensures that substituting one object for another won't break the program.
- **Interface Segregation Principle (ISP):** Clients shouldn't be forced to depend on interfaces they don't use. It's better to have smaller, more specific interfaces tailored to the needs of the clients.
- **Dependency Inversion Principle (DIP):** Here, high-level modules shouldn't depend on low-level modules. Both should depend on abstractions, not concrete details. This encourages decoupling between modules, making the code more flexible.

5. What is an Actuator in Java Full Stack Development?

In Java Full Stack Development, Actuator is a Spring Boot feature that offers ready-to-use endpoints for effectively monitoring and managing applications. These endpoints provide essential details about the application's health, metrics, and environment, facilitating simplified management, especially in production environments.

Conclusion

Java Full Stack Development is one of the most in-demand skill sets in the IT industry. By combining various languages and tools in both frontend, backend, and database, like HTML, CSS, Bootstrap, and Java and MySQL respectively, a Java Full Stack Developer contributes holistically to the process of web application building. Thus, being a full-stack developer can indeed be a thriving career, and these Java full-stack developer interview questions and answers will hopefully give you a head start in that journey. Gain expertise with sought-after IT skills in our software training institute in Chennai.