By OnlineInterviewQuestions.com

ESB interview questions

ESB also was known as Enterprise Service Bus is a programming language that is used to help provide solutions to major administration issues. Due to its rapidity, ease, and effectiveness, many big organizations are looking for candidates with both theoretical knowledge and experience in ESB. Read below some significant and essential **ESB interview questions** to help you get an overview of the subject and also ace your interview:

Q1. What do you understand by Mule?

Mule ESB (also known as Mule) is a lightweight Java-based Enterprise Service Bus (ESB) and an incorporation stage that enables engineers to associate applications rapidly and effectively, empowering them to trade information. Mule ESB empowers simple coordination of existing frameworks, paying little respect to the distinctive advances that the applications use, including JMS, Web Services, JDBC, HTTP, etc.

Q2. What do you understand by ESB?

An Enterprise Service Bus (ESB) is a programming language for engineering especially for middleware that gives major administrations to progressively complex models. For instance, an ESB incorporates the highlights requirements to actualize a service-oriented architecture (SOA). In simple terms, an ESB can be thought of as a component that oversees access to applications and administrations (particularly inheritance variants) to introduce a solitary, straightforward, and steady interface to end-clients through Web-or structures based customer side front end.

Q3. List the different primitives that are constantly used in Mediation.

Some of the commonly used primitives in Mediation include the following:

- Message Filter
- Type Filter
- Endpoint Lookup
- Service Invoke
- Fan – out
- Fan – in
- XSLT
- BO Map

Q4. What do you understand by Shared Context?

It is a setting that is a brief territory, which is made alongside the Service Message Object (SMO) in Mediation Flows. Shared Context is a kind of setting which is available in the SMO. It is essentially utilized when we are utilizing Aggregation process where we have to iterate the BO occasions. Shared Context keeps up Meditation information between Meditation (Fan – Out and Fan – In) primitives. The Content that is available in the common setting BO does not hold on crosswise over Request and Response streams. For example, The Data in the Shared Context, which is utilized in Request stream can't be utilized again in Response stream.

Q5. What do you understand by Transient Context?

Transient Context is used for passing qualities between Mediation primitives inside the present stream, which includes either the demand stream or the reactions stream. The transient setting cannot interface solicitations and reactions and consequently cannot be utilized over. It is used when one needs to save an input before an administration invokes a call. After all the administrations conjure call, the following crude can make another message by consolidating the administration summon reaction and the first message put away in the transient setting.

Q6. What can be used to implement a loop in mediation?

In order to implement a loop into mediation, one needs to use the “Fan – in” and “Fan – out” primitive.

Q7. What method can be applied to change the runtime using mediation primitive?

In order to apply changes to the runtime during mediation primitive, promotable properties such as in ESB training Bangalore is generally utilized. One can always install it during the process of development as well. After which a runtime change can be applied without the need for restarting the server.

Q8. What configurations are required for the implementation of JDBC Adapter?

In order to implement JDBC Adapter, a data source is created that needs to be configured alone with DB. Once the security check is completed then the initialized security is finally authenticated.

Q9. Differentiate between SDO and SMO

SDO is an acronym for Service Data Object, which is a representation of any variable or an object. While on the other hand, SMO is a model that follows a particular pattern for utilizing the SDO objects in order to represent

data messages.

Q10. What do you understand by Mule UMO?

A Mule UMO is a Universal Message Object. It is presently an inheritance term, which was once alluded to as UMO Components are presently alluded to as Service Components. The java and Mule condition factors must be set up effectively for Mule to begin. In the event that one is encountering issues, check the accompanying factors:

- MULE_HOME: ought to be the area of the donkey introduce
- JAVA_HOME: ought to be the area of the JDK
- PATH: ought to have both JAVA_HOME\bin and MULE_HOME\bin in the way.

Check the majority of the above cautiously. A few frameworks with numerous JDK's introduced can finish up with wrong mappings between the PATH and the JAVA_HOME, which will prevent Mule from stacking.

Q11. What methodology is adapted to add jars/ classes into the Mule classpath?

In order to add jars or classes into the Mule classpath, the following procedure is followed:

- Use the MULE_LIB variable (for the most part set in the run content)
- To incorporate JAR file(s) in a Mule class way, announce every needy container document in the MULE_LIB section.
- For spring asset, if the XML bean announcement is put inside a venture, incorporate the task JAR record in the class way as well (i.e., if excluded, Mule will toss a **.xml not found on the class way)

Q12. What do you understand by the term Mule Data Integrator?

Mule has discharged an information integrator apparatus, it is a visual mapping device which underpins level document, java object, XML mappings, and so forth. Coding complex mappings can be dull and hard to keep up; the Mule information integrator with intuitive offices makes constructing and keeping up mappings basic. The mapping is done in overshadowing (modules required) and executed on an information integrator runtime which sits over Mule ESB – this requires a permit.

Q13. Where can one locate the Abstract Mule TestCase?

In order to locate the Abstract Mule TestCase, the 1.4/1.4.1 class characterized in /lib/mule/mule center .jar. is used. The plan provides instructional exercises and inquiries questions are pragmatic and educational. At TekSlate, it offers assets to enable one to learn different IT courses. It also provides both composed material and demos video instructional exercises. For top to bottom learning and pragmatic experience, investigate “Online Mule ESB Training”.

Q14. What do you understand by Web service API?

API is an acronym for Application Programming Interface. It is generally used to write codes by third parties in order to interface with other systems. While, a Web Service is a type of API that constantly operates over HTTP like SOAP, SMTP, etc.

Q15. What are the different strategies involved in Flow Processing?

The different strategies involved in flow processing include:

- Queued Flow Processing Strategy
- Synchronous Flow Processing Strategy
- Thread Per Processing Strategy
- Custom Processing Strategy

Q16. Write a coding to help configure an FTP handler into the Mule ESB

The following code can be used to configure an FTP handler into the Mule ESB ftp_handler-config.xml

```
<?xml version="1.0"encoding="UTF-8"?>
<!DOCTYPE mule-configuration PUBLIC "-//MuleSource //DTD mule-configuration XML V
                                    "http://mule.mulesource.org/dtds/mule-configurati
<mule-configuration id="ftp_handler"version="1.0">
  <description>
   Ftp get to a remote server and place into a local directory on the MULE serve
  </description>
   <!--
  An interceptor is a piece of code that can be configured to execute
   before and/or after an event is received fora component.
   You can define a stack of interceptors that will be executed in sequence.
   You can then configure the stack on your components.
  -->
 <interceptor-stack name="default">
     <interceptor className="org.mule.interceptors.LoggingInterceptor"/>
     <interceptor className="org.mule.interceptors.TimerInterceptor"/>
 </interceptor-stack>
 <!--
    The Mule model initializes and manages your UMO components
 -->
```

```
 <model name="Retrieve_File">
     <!--
         A Mule descriptor defines all the necessary information about how your c
    -->
  <mule-descriptor name="ftpInbound"
      implementation="org.mule.components.simple.BridgeComponent">
      <inbound-router>
          <endpoint address="ftp://mule:mule@localhost/ftp">
     <filter pattern="*.txt"
     className="org.mule.providers.file.filters.FilenameWildcardFilter"/>
     <properties>
         <property name="binary"value="false"/>
         <property name="pollingFrequency"value="1000"/>
         <property name="filename"value="document.txt"/>
         <property name="outputPattern"value="FtpFile-${DATE}.done"/>
     </properties>
 </endpoint>
  </inbound-router>
  <outbound-router>
      <router className="org.mule.routing.outbound.OutboundPassThroughRouter">
          <endpoint address="file:///C:/MULE/inbound"/>
      </router>
   </outbound-router>
   <!--
      Here we tell this component to use the interceptor stack defined above
   -->
   <interceptor name="default"/>
  </mule-descriptor>
  </model>
</mule-configuration>
```