**1. What is the directory structure in JBoss?**

The following directories are available after JBoss is installed.

* modules
* bundles
* domain
* standalone
* appclient
* bin
* docs
* welcome-content

**2. What are the logging levels available?**

There is five possible level:

* FATAL
* ERROR
* WARN
* INFO
* DEBUG

**3. Which component is responsible for handling clustering?**

JBoss clustering is on top of JGroups toolkit which helps to create, delete, membership detection, notification, etc. in the cluster.

**4. How to install JBoss on Linux server?**

JBoss installation is very straightforward. You need to download the desired version from JBoss official download page in zip or gz format.

Once downloaded, just extract the file to the location you want to install. If you downloaded zip format, then you can use unzip command to extract it.

unzip jboss-as-7.1.1.Final.zip

**5. What’s the default port to access Administration Console in JBoss 7?**

9990 is the default port. If it’s installed on **server1** then you need to access like:

<http://server1:9990/admin-console>

**6. What must be done to access Admin Console?**

The user must be created under “ManagementRealm” to have console operational. To create the user, you can go to bin folder and execute add-user.sh script.

**7. How to start JBoss in standalone mode?**

Go to bin folder where JBoss is installed and start with the following command.

./standalone.sh

**8. How to increase Java Heap Memory in JBoss 7?**

Heap Memory can be increased in a respective conf file. To increase memory for standalone;

* Go to bin folder
* Edit the standalone.conf file and look for “JAVA\_OPTS=” argument line
* The default configuration will have minimum 64 MB and maximum 512 MB. You can increase to the desired value.

Xms – specify the minimum heap size

Xmx – specify the maximum heap size

In a similar way, you can adjust the memory for a domain in domain.conf file

**9. What is the difference between standalone and domain mode?**

**Standalone** mode is single JVM process where every JBoss server has its configuration. If you just need one JVM or development environment, then standalone would be perfect.

**Domain** mode may have multiple servers where all configuration is managed centralized and often used in production environment.

**10. Can you create a cluster in standalone mode?**

Yes, clustering is possible in standalone mode. However, an application must be deployed on each server/JVM in standalone mode.

**11. What is difference between <validate-on-match> and <background-validation>?**

<**validate-on-match**> validate the database connection every time, and if a connection is not valid, it will write a warning in the logs.

Having “validate-on-match” configured may have a little high load on the database as it may create lots of requests.

<**background-validation**> validate the connection periodically based on what frequency is configured for “background-validation-millis”. The default configuration is set to zero means disabled.

Having “ background-validation” set to true will create fewer database connections and it’s side-effects would be not detecting immediately if dead connections.

**12. What module needed to integrate Apache with JBoss?**

There are two modules you can use to connect JBoss with Apache.

1. mod\_proxy
2. mod\_jk

**13. What are the file types you can deploy in JBoss?**

You can deploy almost any kind of Java/J2EE application, and it supports the following file format.

* WAR – Web application archive
* SAR – Service archive
* JAR – Java Archive
* EAR – Enterprise application archive

**14. How can you deploy an application?**

There are three possible ways to deploy an application in JBoss application server.

1. Admin Console – you can deploy the necessary application files through the administration console.
2. Auto-deploy – leverage file system deployment scanner to auto deploy files from deployments folder.
3. Automation – use automation tool/ant/scripting to deploy an application.

**15. What marker file type is required to instruct JBoss to deploy?**

.dodeploy file suffix is needed for JBoss to deploy or redeploy an application. For ex:

myfirstapplication.war.dpdeploy

**16. What are the important types available for marker file deployment?**

* .dodeploy – instruct to deploy
* .deployed – indicate the file is deployed
* .pending – deployment is still pending
* .undeployed – confirmation that application is undeployed
* .failed – deployment is failed for some reason
* .skipdeploy – instruct JBoss to ignore the files for auto-deployment

**17. What does mgmt-user.properties contain?**

All admin console users and password (encrypted) are stored in mgmt.-user.properties file.

################# 2 #########################

|  |
| --- |
| What are the components that are used in Jboss Cache? Jboss cache is used as a replicated and transactional cache. It is said like that due to the fact that multiple Jboss cache instances can be distributed in java virtual machine or each virtual machine will have its own copy of it. These cache can reside on the same machine or different machines that is on the network. This keeps the replicated data across the network in case any failure arises. The transactional property is due to the fact that user can create and configure a compliant transaction manager and make their own cache operations. If the cache is not replicated then it is running in a local mode and it has two components that are involved:  **1. Generic cache :** It has internal implementation and the method can be used as org.jboss.cache.TreeCache **2. POJO cache :** Is also implemented internally by using the method as: org.jboss.cache.aop.PojoCache. It extends the functionality of Tree. **3. TreeCache :** Is used to represent a tree like cache that provides easy replication and transaction properties to extend the functionality of the object cache. It provides a transparent mapping tool to map it to the internal cache. |
| What are the steps that are involved in deploying Jboss cache using Mbean service? JBoss cache provides MBean service that can be registered with the environment to allow the cache access using JMX. The steps that are involved in deploying JBoss cache that is using Mbean service is that:  1. First copy the configuration xml file and deploy it in the directory that is given by the JBoss cache. The configuration files from all other sections are present there only. 2. Check the etc/META-INF directory for the configuration files that are for different cache modes and that can be used to deploy the JBoss cache. 3. Get the command to invoke the right service like create, start, stop, and destroy operations t be performed on the files given in the directory. 4. Look for the details of the cache’s current state to see the process which can be executed or contained before any other. 5. Cache configuration files are checked and necessary changes are made when the cache is already running. |
| What is the procedure to access the JBoss Cache MBean service? To access the MBean service the JBoss sets the server using many header files like Proxy files, Server locater files, cache configuration files etc. The sample code given below explains the procedure.  org.jboss.mx.util.MBeanServerLocator; org.jboss.mx.util.MBeanProxyExt; org.jboss.cache.TreeCacheMBean; javax.management.MBeanServer;  MBeanServer server; TreeCacheMBean cache; // These are the header files that are given above. It consists of all the files like server files, MBean service files, cache configuration files, etc. public init() throws Exception // This is the start function by which the process starts { try // This is the exception handling block to find the errors at runtime {         server = MBeanServerLocator.locateJBoss();        cache = (TreeCacheMBean) MBeanProxyExt.create(TreeCacheMBean.class, "jboss.cache:service=TreeCac  } catch (Exception ex) {         // handle exception  } } |
| What are the main directories provided in JBoss? The main directories that are provided in JBoss directory structure are as follows:  **1. bin –** It consists of system executable and normal files that are for execution purpose. It consists of binary files for every program that is installed on the system. **2. client –** It consists of all the configuration files that are stored and jar files that can be required by the Java client application. **3. docs –** It consists of the XML documents with the DTDs for the reference. These documents provide the basic specification related to the installation and deployment of JBoss. **4. lib –** This consists of jar files that are required to run the JBoss microkernel. This is directory is not for any user as it consists of the kernel and system libraries. **5. server –** This consists of different server configuration files. The files are in the script format. |
| What are the different types of services provided by JBoss server? The services of the server are based on the type of files that are provided and kept by the server. The server provides the service from basic to advanced and they are classified as:  **1. Core services :** These are the services that provide scripts to start the server and provide the basic functionality of the start-up scripts. **2. Logging Service :** These are the services that are used for logging on the server and using the configuration setting to modify and create it. **3. Security Service :** These are the services that all the security domain information to be stored in the file login config.xml files and used to list the named security domain features. It specifies all the separate modules that are used for authentication purpose in the domain. **4. Additional Services :** These are the services that are hot-deployable and directly deployed in the directory. They consist of XML descriptor files and other resource files. |
| What are the core services provided by the JBoss server? The core services are specified in the configuration files that are used by the JBoss. This file lies in the configuration folder. The file is called as service.xml and it is the first file that starts when the server gets started. This file consists of the configuration MBeans for various services including the logging, security, etc. The example of the file is given as:  <!--  name="jboss:service=JNDIView" xmbean-dd="resource:xmdesc/JNDIView-xmbean.xml">  -->  This way a service can be stopped and modified and this effect when the server gets restarted. It doesn’t list the services that are removed in the service management console. |
| What are the Logging Services provided by the JBoss server? The logging server allows the user to log into the server and save its logging details which might be used again. The package that is used for logging is log4j. The logging onto a server is controlled from a central file that resides in the configuration file that is managed separately. The file name I given as log4j.xml and this file defines a set of appending files that also considered as log files. The file follows a specific format that should go with the message and there is also a filtering option that is used to filter out the details that are not required. The output of this file gets to both the console and a log file that is also called as server log given in the log directory. Logging services are very important to manage as it keeps the defined data on the server of people who are logging in. |
| What are the different types of log levels that are used in logging services in JBoss? There are four different types of log levels that are used to show the services that are present inside the logging in JBoss. The log levels are as follows:  **1. Debug :** In this level the file gets debugged for possible errors. This can either be automatic or manual as it doesn’t affect the file in any way. **2. Info :** At this level the complete detail information is provided about the process that is running in the background. It consists of details like name of the process, CPU usage, etc. **3. Warn :** At this level the warning that has come in the file that might hinder the performance on some systems are given and it is displayed with some critical signs to aware the user to see the warning and take necessary actions if required. **4. Error :** At this level during the compilation of the error there are some errors shown that can be syntactical error or logical errors. These errors have to be fixed before completing the compilation process. |
| What is meant by logging threshold in JBoss? The logging threshold is given as INFO as it is a level on which all the informational messages, warning messages and error messages on the console are produced and shown, but not the debug messages. It is managed through the file as there is no threshold set for the server.log file. The generated logging messages will be stored in that section to provide all the information from one place. The log files are always checked in case of any errors that are produced due to some information on the debug messages that can help in tracking down the problem. The logging threshold allows the debug messages to be displayed with the complete information about the log files. The logging limits are set for individual categories. |
| Write a program to show the change in level in JBoss server. If the change in the level has to be set from INFO to DEBUG level then the output from the container-managed persistence engine has to be changed to the DEBUG level. A redirection takes place in separate files and a log is maintained with the name cmp.log to compare the files later on. The code that is written has to be added in log4j.xml file:  <appender name="CMP" class="org.jboss.logging.appender.RollingFileAppender"> <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/> <param name="File" value="${jboss.server.home.dir}/log/cmp.log"/> <param name="Append" value="false"/> <param name="MaxFileSize" value="500KB"/> <param name="MaxBackupIndex" value="1"/> <layout class="org.apache.log4j.PatternLayout"> <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/> </layout> </appender> <category name="org.jboss.ejb.plugins.cmp"> <priority value="DEBUG" /> <appender-ref ref="CMP"/> </category> |
| What is the role of an appender in logging when used in JBoss? The file that is created using appender specifies the use of it as logger for the package org.jboss.ejb.plugins.cmp. This makes it easy for the server to look at CMP and compare the two log files. The files are used to produce a new log file everyday and don’t produce any file even there are changes made every time. The service has to be restarted and during the restart there is no server file gets written or done. The older files consist of the dates that are written and added to the name of it. The log directory consists of HTTP request logs that are produced by the web container. |
| What are the Security Services provided by the JBoss server? Security Service consists of the domain information that is stored in the file in the login config.xml file and it has all the named security domains that specify the number of login modules that are used for authentication purposes in the domain. The security for the application can be used till it specifies the name of the domain to be used in specific JBoss deployment descriptors. The files that are to be considered are jboss.xml and jboss-web.xml. This is used to secure the JMX console application that comes with the JBoss. Every aspect of the server is controlled through the JMX console. The applications are kept password protected and any remote user can completely control the server without having any detailed information. |
| Write a program to show the security domain feature to cover the application in JBoss. The security domain provides features through which the server can be protected. This is done using the file jboss-web.xml this can be used to deploy in jmx-console.war/WEB-INF/ directory. JBoss uses JAAS services to provide the pluggable authorization modules and the code that is given below explains it as well:  <jboss-web> <security-domain>java:/jaas/jmx-console</security-domain> </jboss-web>  This links to the security domain where the web application can be connected and security policies can be applied to it. URLs are trying to protect the server configuration that is allowed to configure the web.xml file in the same directory and the security constraints are put together to provide the user name and password as well. |
| Write a program to show the security constraints that show the role of the user and the admin in using the server. The security constraints that are have restrictive access to the HTML JMX console and it allows few users with specified roles to be the JBossAdmin. The modification in the roles can be done and the file can be edited to modify the security domain element to enable the secured feature of the services. The code is as follows:  <security-constraint> <web-resource-collection> <web-resource-name>HtmlAdaptor</web-resource-name> <description> </description> <url-pattern>/\*</url-pattern> <http-method>GET</http-method> <http-method>POST</http-method> </web-resource-collection> <auth-constraint> <role-name>JBossAdmin</role-name> </auth-constraint> </security-constraint> |
| Write a program to show how to set the username and password using the security domain in JBoss. The username and password is required for the security purpose to produce at the logging time. The user name and passwords gets stored in the JMX console security domain that manages the application. The configuration file can be provided using the login-config.xml. The program that is being written in which the modification can be done is shown below:  <application-policy name="jmx-console"> <authentication> <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule" flag="required"> <module-option name="usersProperties"> props/jmx-console-users.properties </module-option> <module-option name="rolesProperties"> props/jmx-console-roles.properties </module-option> </login-module> </authentication> </application-policy>  This is the configuration file that uses simple file based security policy and found in the config directory. It consists of the server configuration and the properties that is required to assign the user about the files. |
| What are the additional services that are provided with the JBoss server? Additional services are used as non-core, hot-deployable services that are added to the deploy directory and it consists of XML descriptor files. It is consisting of the following properties and files that are used as:  Service.xml or the SAR (JBoss Service archive) file that consists of both XML descriptor and additional resources that are required by the service provider all in a single archive file. There is given the detailed information on all these services that can be found in JBoss that provides the information on server internals and use implementation services like JTA and uses the JCA (J2EE connector architecture) |
| What is the purpose of JMS and Message-Driven Beans in JBoss? JMS is used for messaging system and it also consists of message driven beans that is used in JBoss. This consists of the MDB concepts that are used in the jboss-build.xml file. This JSM (Java message service) provides a middleware orientation to send message between two or more clients. JMS is a part of Java Platform that is used to develop the application that is under the java platform. This also allows the application components to be based on the Java Enterprise Edition (JEE). It handles the creation, sending, receiving, and reading the messages. It also allows the communication between many different components that are distributed in many parts and are loosely coupled, reliable and asynchronous message passing. |
| What is the process of compiling and packaging the MDB files? To compile the files of MDB (Message driven beans) first there need to be compilation process that will take place invoking the compile-mdb that targets the simplemessage directory. It produce the targets to be made archives for the bean and the client uses the file that is in the JAR directory. The commands that are used to compile the files are as follows:  ant -f jboss-build.xml package-mdb ant -f jboss-build.xml package-mdb-client ant -f jboss-build.xml assemble-mdb  These files that are used and made by the command gets combined in the directory and that directory will be called as the package. It consists of the deployment descriptors that are used in the jar directory containing all the archives that are produced by the build of the MDB files. |
| Explain with an example the implementation of source queue for MDB. The implementation of the source queue for MDB is used to be the container specific file that consists of information regarding the queue like queue name for MDB that resides in jboss.xml and the source file that is used with it remains in the queue till the message is not being transferred by the server. The server executes the file and keeps a log file to maintain all the information. The modification can be done by using the source queue. If any additional information, be required using the source queue then message driven beans allow the use of extra parameters like destination, name, etc. to describe the objects. The code is shown below:  <jboss> <enterprise-beans> <message-driven> <ejb-name>SimpleMessageBean</ejb-name> <destination-jndi-name>queue/MyQueue</destination-jndi-name> |