**中软国际考题-笔试**

1. 单选题

1、使用JNDI时一般需要构建哪个Context（B）

A. InitialContext B.NamingContext C.IoCContext D.JavaEEContext

2、下列哪种方法可以获取数据源（C）

A.通过Http

B.通过ftp

C.JNDI

D.通过Connection对象

3、JNDI可用于如下哪些应用场景（C）

A.配置信息存储

B.异步消息发送

C.数据库连接池查找

D.远程对象查找

4、下面哪种服务不是JNDI应用范围（A）

A.Servlet B.EJB C.JDBC D.JMS

5、下面哪个类是JNDI的驱动类？（A）

A．java.naming.factory.initial；

B．java.naming.factory.DriverManager；

C．java.naming.driver.initial；

D．java.naming.jndi.Driver;

6、哪个不是JNDI提供的程序包：D

1. Javax.naming
2. Javax.naming.directory
3. Javax.naming.event

Javax.naming.app

7、常用JNDI服务提供者连接工长说法错误的是（）

a) Filesystem: Com.sun.jndi.fscontext.FSContextFactoryLDAPv3

b) NDS: com.novell.naming.service.nds.NdsInitialContextFactory

c) RMI registry: com.sun.jndi.rmi.registry.RegistryContextFactory

d) JBOSS名字服务提供者：weblogic.jndi.WLInitialContextFactory

8. 什么方法用于查找目录对象（A）

a) lookup

b) findByDirectory

c) get

d) list

9. 以下哪个包提供了通过javax.naming和有关包动态增加对访问命名和目录服务的支持。(D)

a) Javax.naming.directory

b) Javax.naming.ldap

c) Javax.naming.extens：

d) Javax.naming.spi

10.Web Services 标准和规范有那些？请从下面的答案中选择。

A, ARP、SNMP、TCP/IP、UDP;

B, COBRA、RMI、UDDI、WS;

C, FTP、HTTP、SNMP、SOAP;

D, SOAP、WDSL、UDDI、WS-Security。

答案：\_\_\_\_\_\_\_\_D

11. 以下哪种技术可以取得数据源。C

A.通过Http

B.通过ftp

C.JNDI

D.通过Connection对象

1. 多选题
2. 以下哪些是JNDI提供的常用包？（A、B、C、D）

A．javax.naming

B. javax.naming.directory

C. javax.naming.event

1. javax.naming.ldap
2. JNDI可以访问的目录和服务有哪些？（A、B、C、D、E、F）
3. DNS B.XNam C. Novell
4. LDAP E. 文件系统
5. Windows注册表

3、下列哪些可以用JNDI访问？（A、B、C、D）

A. J2EE中的JNDITree

B. 文件系统

C. LDAP目录服务

D. DNS域名服务器

4、关于JNDI，下列说法正确的是（ABCDE）

A．包含大量命名和目录服务，可以使用相同API 调用访问任何命名或目录服务。

B．可以同时连接多个命名和目录服务。

C．允许把名称同JAVA 对象或资源关联起来，不必知道对象或资源的物理ID。

D．使用通用接口访问不同种类的目录服务

E．使得开发人员能够集中使用和实现一种类型的命名或目录服务客户API 上。

5、下列说法正确的是:（ ABCD ）

1. JNDI：提供了一种统一的方式，可以在本地或网络上查找和访问服务。
2. 各种服务在命名服务器上注册一个名称，需要使用服务的应用程序通过JNDI找到对应服务就可以使用。
3. JNDI：可以获得访问数据相关的对象，从而操作数据。本地不需要驱动，也不需要知道数据在哪里，所有需要访问数据库的都可以用
4. JNDI能注册各种各样可用的服务，使用JDBC实现的数据访问服务是其中之一

6、 列举自带jndi的实现 (ABCD)

a) LDAP，

b) CORBA，

c) RMI

d) DNS

e) JNI

1. 填空题（每空1分，共10分）
2. \_\_\_\_\_\_\_\_\_\_包通过javax.naming和有关包动态增加对访问命名和目录服务的支持。这个包是为有兴趣创建服务提供者的开发者提供的。 (javax.naming.spi)
3. JNDI是Java­­\_\_\_\_\_\_和\_\_\_\_\_\_接口的简称，从一开始就一直是Java平台企业版的核心技术之一。(命名、目录)
4. JNDI是Java 命名和目录接口，它的英文全称是\_\_\_\_\_\_\_\_\_\_\_\_\_\_。(Java Naming and Directory Interface)
5. JNDI是一组帮助做多个\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_接口的API。(命名、目录服务)
6. 在Jboss 中要自定义JNDI 名称，可以使用  和 注释（@LocalBinding @RemoteBinding ）
7. JNDI的结构由一个 API 和一个 SPI 组成，Java应用程序实用JNDI API访问[各种各样](https://www.baidu.com/s?wd=%E5%90%84%E7%A7%8D%E5%90%84%E6%A0%B7&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/u011878202/article/details/_blank)的命名和目录服务。
8. JNDI为开发人员提供了查找和访问各种 命名 和 目录服务 的 通用 、 统一 的方式。
9. 使用RMI实现JNDI对象绑定到服务器上，对象必须继承/实现\_\_\_\_\_\_\_\_。对象要在网络中传输必须要序列化，所以继承/实现\_\_\_\_\_\_\_\_ (Remote， Serializable)
10. 判断题
11. JNDI是一组在Java应用中访问命名和目录服务的API.命名服务将名称和对象联系起来，使得我们可以用名称访问对象。目录服务是一种命名服务，在这种服务里，对象不但有名称，还有属性。（ T ）
12. 通过在Tomcat下配置JNDI数据源，能够有效的提高数据库的访问效率，提高系统的并发能力。（ T ）
13. JNDI这个API提供了五个包，我们实际上只需要使用其中的几个包就可以。（ T ）
14. 可以调用Context的find方法查找Java对象。（ F ）
15. 将一个Context对象绑定到另外一个Context对象中，这就形成了一种父子级联关系。（ T ）
16. JNDI中的目录（Directory）是指将一个对象的所有属性信息保存到一个容器环境中。（ T ）
17. JNDI的目录（Directory）与命名（Naming）一般不能结合在一起使用。（ F ）
18. 使用Resource来表名一个JNDI数据源。（ T ）
19. maxActive用于指定连接池中处于活动状态的数据库最大连接数（ T ）
20. 很多J2EE技术，包括EJB都依靠JNDI来组织和定位实体。（ T ）
21. javax.naming的包包中提供Context接口，使用combind命令来绑定对象。（ F ）
22. 目录服务是一种命名服务，在这种服务里，对象不但有名称，还有属性。 （ T ）
23. 在JNDI中，在目录结构中的每一个结点称为context。每一个JNDI名字都是相对于context的。（ T ）
24. JNDI避免了程序与数据库之间的紧耦合，使应用更加易于配置、易于部署（ T ）
25. JNDI需要程序员在开发时通过代码对数据库url，username，password等参数进行管理（ F ）
26. JNDI在满足了数据源配置的要求的基础上。还进一步扩充了作用：全部与系统外部的资源的引用，都能够通过JNDI定义和引用。（T）
27. JNDI的结构由一个API和一个SPI组成，Java应用程序实用JNDI API访问各种各样的命名和目录服务。（ T ）
28. 命名服务是将名称与对象相关联。这种关联被称为绑定。一组这样的绑定被称为上下文，Jndi上下文可以用来查找，捆绑/解除捆绑，创建或者破坏绑定名称操作在JNDI中，上下文是使用javax.naming.Context 接口来表示的，而这个接口也正是与命名服务进行交互的主要接口。（T）
29. JNDI包含大量命名和目录服务，可以使用相同API 调用访问任何命名或目录服务。(T)
30. JNDI可以同时只能连接一个个命名和目录服务。（F）
31. 允许把名称同JAVA 对象或资源关联起来，不必知道对象或资源的物理ID。（T）
32. 使用通用接口访问不同种类的目录服务。（T）
33. 使得开发人员能够集中使用和实现一种类型的命名或目录服务客户API 上。(T)

五、简答题

1、简介JNDI程序包的结构

答案：

javax.naming：命名操作；

javax.naming.directory：目录操作；

javax.naming.event：在命名目录服务器中请求事件通知；

javax.naming.ldap：提供LDAP支持；

javax.naming.spi：允许动态插入不同实现。

利用JNDI的命名与服务功能来满足企业级APIs对命名与服务的访问，诸如EJBs、JMS、JDBC 以及IIOP上的RMI通过JNDI来使用CORBA的命名服务。

2、请简单描述JNDI的优点（5分）

答案： JNDI(Java Naming and Directory Interface)是SUN公司提供的一种标准的Java命名系统接口。JNDI包含了大量的命名和目录服务，使用通用接口来访问不同种类的服务；  
可以同时连接到多个命名或目录服务上；建立起逻辑关联，允许把名称同Java对象或资源关联起来，而不必指导对象或资源的物理ID。JNDI可访问的现有的目录及服务有：DNS、XNam 、Novell目录服务、LDAP(Lightweight Directory Access Protocol 轻型目录访问协议)、 CORBA对象服务、文件系统、Windows XP/2000/NT/Me/9x的注册表、RMI、DSML v1&v2、NIS。

3、常用的JNDI操作都有哪些?

答案：

void bind(String sName,Object object);――绑定：把名称同对象关联的过程

void rebind(String sName,Object object);――重新绑定：用来把对象同一个已经存在的名称重新绑定

void unbind(String sName);――释放：用来把对象从目录中释放出来

void lookup(String sName,Object object);――查找：返回目录总的一个对象

void rename(String sOldName,String sNewName);――重命名：用来修改对象名称绑定的名称

NamingEnumeration listBinding(String sName);――清单：返回绑定在特定上下文中对象的清单列表

NamingEnumeration list(String sName);

4、JDBC与JNDI这两种连接方式有什么区别？（5分）

答案：Java Database Connectivity (JDBC)是一个标准的Java API，它由一组类和接口组成，Java应用程序开发人员使用它来访问数据库和执行SQL语句，直接连访问数据库。

JNDI(Java Name Directory Interface，Java命名和目录接口)，它不仅仅是进行数据库定位的，它是给当前应用服务器所管理的所有资源一个唯一的标识，包括数据库，网页，文件，连接池等等。JNDI提供了一种统一的方式，可以用在网络上查找和访问服务。通过指定一个资源名称，该名称对应于数据库或命名服务中的一个记录，同时返回数据库连接建立所必须的信息。

5、解释什么是JNDI？

答案：Java Naming and Directory Interface (JNDI)

JNDI API被用于执行名字和目录服务。它提供了一致的模型来存取和操作企业级的资源如DNS和LDAP，本地文件系统，后者在应用服务器中的对象。在JNDI中，在目录结构中的每一个结点称为context。每一个JNDI名字都是相对于context的。应用可以通过这个初始化的context经有这个目录树来定位它所需要的资源或对象。

6、介绍一下JNDI程序包的结构

答案：javax.naming：命名操作；

javax.naming.directory：目录操作；

javax.naming.event：在命名目录服务器中请求事件通知；

javax.naming.ldap：提供LDAP支持；

javax.naming.spi：允许动态插入不同实现。

7、什么叫JNDI？谈谈你对JNDI的理解。

答：JNDI全称 Java Naming and Directory Interface   
JNDI是Java平台的一个标准扩展，提供了一组接口、类和关于命名空间的概念。如同其它非常多Java技术一样，JDNI是provider-based的技术，暴露了一个API和一个服务供应接口（SPI）。这意味着不论什么基于名字的技术都能通过JNDI而提供服务。仅仅要JNDI支持这项技术。

JNDI眼下所支持的技术包含LDAP、CORBA Common Object Service（COS）名字服务、RMI、NDS、DNS、Windows注冊表等等。非常多J2EE技术，包含EJB都依靠JNDI来组织和定位实体。   
JDNI通过绑定的概念将对象和名称联系起来。

在一个文件系统中。文件名称被绑定给文件。在DNS中，一个IP地址绑定一个URL。在文件夹服务中。一个对象名被绑定给一个对象实体。   
JNDI中的一组绑定作为上下文来引用。每一个上下文暴露的一组操作是一致的。比如，每一个上下文提供了一个查找操作。返回指定名字的对应对象。每一个上下文都提供了绑定和撤除绑定名字到某个对象的操作。

JNDI使用通用的方式来暴露命名空间，即使用分层上下文以及使用同样命名语法的子上下文。

8、INITIAL\_CONTEXT\_FACTORY 常量代表的意义

常量保存用来指定要使用的初始上下文工厂的环境属性名称。该属性的值应该是将创建初始上下文的工厂类的完全限定类名称。此属性可以在传递给初始上下文构造方法的环境参数、applet 参数、系统属性或应用程序资源文件中指定。

9.PROVIDER\_URL常量代表的意义

PROVIDER\_URL该常量保存用来指定要使用的服务提供者配置信息的环境属性名称。该属性的值应该包含一个 URL 字符串（例如 "ldap://somehost:389"）。此属性可以在环境、applet 参数、系统属性或资源文件中指定。如果没有在任何这些源中指定该属性，则由服务提供者确定默认配置

10.URL\_PKG\_PREFIXES常量代表的意义

URL\_PKG\_PREFIXES该常量保存用来指定加载 URL 上下文工厂时要使用的包前缀列表的环境属性名称。该属性的值应该是工厂类名称以冒号分隔的包前缀列表，该工厂类将创建一个 URL 上下文工厂。此属性可以在环境、applet 参数、系统属性或者一个或多个资源文件中指定。前缀com.sun.jndi.url 总是被追加到可能为空的包前缀列表中

11.简述JNDI程序包

JNDI程序包：   
javax.naming：命名操作；   
javax.naming.directory：目录操作；   
javax.naming.event：在命名目录服务器中请求事件通知；   
javax.naming.ldap：提供LDAP支持；   
javax.naming.spi：允许动态插入不同实现。

六、程序实现题

1、编写Tomcat中创建JNDI数据源的主要代码以及在Java中使用该数据源的主要代码

|  |
| --- |
| <Context path="/test" docBase="/test">  <Resource name="jdbc/drp"  auth="Container"  type="javax.sql.DataSource"  driverClassName="com.mysql.jdbc.Driver"  url="jdbc:mysql://localhost:3306/drp"  username="root" password="123456"  maxActive="20" maxIdle="10"  maxWait="10000"/>  </Context>  ------------------------------------------------------------------------------  //初始化查找命名空间  Context ctx = new InitialContext();  //InitialContext ctx = new InitialContext();亦可  //找到DataSource,对名称进行定位  DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/drp");  //取出连接  conn = ds.getConnection();  out.println(conn);  out.println(conn.isClosed()); |

2、基于MySql数据库，配置Tomcat数据源，并且通过代码实现对数据源的查找，返回数据库连接。（10分）

|  |
| --- |
| 在tomcat根目录\conf\context.xml文件（没有直接创建也可以）或者在JSP项目WebRoot目录下的META-INF目录中创建一个context.xml文件，添加Context节点，如下：  <Context>  <Resource name="jdbc/test" auth="Application" type="javax.sql.DataSource"  maxActive="100" maxIdle="30" maxWait="10000" username="sa" password="1234"  driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver" url="jdbc:sqlserver://192.168.2.254:1433;DatabaseName=test" />  </Context>  import java.lang.reflect.\*;  import java.sql.\*;  import java.util.\*;  import javax.naming.Context;  import javax.naming.InitialContext;  import javax.naming.NamingException;  import javax.sql.DataSource;  public class SqlHelper {    private static Connection getConnection() throws ClassNotFoundException,  SQLException {  Connection conn = null;  try {  Context ctx = new InitialContext();  // 获取与逻辑名相关联的数据源对象  DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/test");  conn = ds.getConnection();  } catch (NamingException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  return conn;  } |

3、请根据要求编写JNDI配置及代码，实现数据库Mysql的连接。

配置文件

要求：

* 数据库地址为：localhost:3306/DemoDB
* 用户名密码：root/root

根据以上要求完成以下配置context.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Context>  <Resource name="mysqlDataSource" type="javax.sql.DataSource"  driverClassName="com.mysql.jdbc.Driver"  url="jdbc:mysql://localhost:3306/shop?useUnicode=true&amp;characterEncoding=utf-8"  username="root" password="root" maxActive="50" maxIdle="10" maxWait="60000"  removeAbandoned="true" removeAbandonedTimeOut="100000" />  </Context> |

根据该配置编写Java类DBUtil，该程序的getConnection方法返回数据库连接Connection

|  |
| --- |
| public class DBUtil{  public Connection getConnection() throws ClassNotFoundException,  SQLException {    try {  Context ctx = new InitialContext();  DataSource ds = (DataSource) ctx  .lookup("java:comp/env/mysqlDataSource");  connection = ds.getConnection();  } catch (Exception e) {  e.printStackTrace();  }  return connection;  }  } |

4、常用的JNDI操作都有哪些?

|  |
| --- |
| void bind(String sName,Object object);――绑定：把名称同对象关联的过程  void rebind(String sName,Object object);――重新绑定：用来把对象同一个已经存在的名称重新绑定  void unbind(String sName);――释放：用来把对象从目录中释放出来  void lookup(String sName,Object object);――查找：返回目录总的一个对象  void rename(String sOldName,String sNewName);――重命名：用来修改对象名称绑定的名称  NamingEnumeration listBinding(String sName);――清单：返回绑定在特定上下文中对象的清单列表  NamingEnumeration list(String sName); |

5、用代码演示如何配置数据源及在程序中使用?

|  |
| --- |
| 配置数据源    在程序中引用数据源： |

6、编写一个连接了本地的文件系统的例子，

需要将fscontext.jar和providerutil.jar加载到classpath下（10分）

|  |
| --- |
| public class FSSPTest {  public static void main(String[] args) throws NamingException {  // TODO Auto-generated method stub  Hashtable<String,String> env = new Hashtable<>();  //指明初始化的factory是我们下载的jar包中的RefFSContextFactory  env.put(Context.INITIAL\_CONTEXT\_FACTORY,  "com.sun.jndi.fscontext.RefFSContextFactory");  //指明Context的初始URL，这里我们的是C盘  env.put(Context.PROVIDER\_URL,"file:///c:/");  Context ctx = new InitialContext(env);  //在C盘下创建要给文件夹名为JesminDir  ctx.createSubcontext("JesminDir");  //在C盘下定位myFile文件  File f = (File) ctx.lookup("myFile");  System.out.println(f);  //列出当前context下的所有元素的名称和类型(包括文件夹和文件)  NamingEnumeration list = ctx.list("/");  while (list.hasMore()) {  NameClassPair nc = (NameClassPair)list.next();  System.out.println(nc);  }  }  } |

7、在使用InitialContext去获取[jboss](https://www.baidu.com/s?wd=jboss&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "_blank)中Ejb对象。创建cxt对象

Hashtable<String, String> c = new Hashtable<String, String>();

c.put("java.naming.provider.url", url);

c.put("java.naming.factory.initial",

"org.jnp.interfaces.NamingContextFactory");

c.put("java.naming.factory.url.pkgs",

"org.jboss.naming:org.jnp.interfaces");

cxt = new InitialContext(c)

8、已知代码， 打印清单列表。

NamingEnumeration namEnumList = ctxt.listBinding(“tx”); 如何获取所有tx下所有的对象名字和类的名字

while ( namEnumList.hasMore() ) {  
Binding bnd = (Binding) namEnumList.next();  
String sObjName = bnd.getName();  
String sClassName = bnd.getClassName();  
SomeObject objLocal = (SomeObject) bnd.getObject();  
}