## 1)JNDI

JNDI ---- Java 命名与目录接口（Java Naming and Directory Interface）.

在J2EE容器中配置JNDI参数，定义一个数据源，也就是JDBC引用参数，给这个数据源设置一个名称；然后，在程序中，通过数据源名称引用数据源从而访问后台数据库。

以下是连接mysql数据库的数据源配置：

Xml代码

1.<Resource

2. name="jdbc/test"

3. type="javax.sql.DataSource"

4. password="bb"

5. driverClassName="com.mysql.jdbc.Driver"

6. maxIdle="2"

7. maxWait="50"

8. username="root"

9. url="jdbc:mysql://localhost:3306/test"

10. maxActive="4"/>

<Resource name="jdbc/test" type="javax.sql.DataSource" password="bb" driverClassName="com.mysql.jdbc.Driver" maxIdle="2" maxWait="50" username="root" url="jdbc:mysql://localhost:3306/test" maxActive="4"/>

### (2) 配置方法(tomcat)

1.第一种：

在App工程\META-INF\context.xml中配置数据源.

2.第二种

配置数据源.

3.第三种

在Tomcat目录\conf\server.xml 中配置数据源,

然后App工程下的context.xml（或者在Tomcat目录\conf\localhost\工程名.xml）链接数据源：

Xml代码

1.<ResourceLink

2. global="jdbc/test"

3. name="jdbc/test"

4. type="javax.sql.DataSource"/>

<ResourceLink global="jdbc/test" name="jdbc/test" type="javax.sql.DataSource"/>

4.第四种

在Tomcat目录\conf\context.xml中配置数据源。

第一，二种是是app工程独有数据源配置方式。

第三，四种是全局数据源配置方式。

### (3) 通过JNDI取数据源

Java代码

1.Context initContext = new InitialContext();

2.Context envContext = (Context)initContext.lookup("java:/comp/env");

3.DataSource ds = (DataSource)envContext.lookup("jdbc/test");

4.Connection conn = ds.getConnection();

Context initContext = new InitialContext(); Context envContext = (Context)initContext.lookup("java:/comp/env"); DataSource ds = (DataSource)envContext.lookup("jdbc/test"); Connection conn = ds.getConnection();

### (4) spring 访问 jndi

Spring中对于JNDI的访问，提供了便捷的方法，在Spring的org.springframework.jndi包中包含了所有的类。其中提供了一下核心类：

（1）JndiTemplate：它是这个包的核心类，用来简化对JNDI的操作。它提供了对于lookup、bind方法。同时，允许JndiCallback接口的实现在其提供的JNDI上下文中执行任何操作。

使用JndiTemplate实现InitialContext环境变量的配置

<bean id="queueTarget" class="org.springframework.jndi.JndiObjectTargetSource">

<property name="jndiName">

<value>queue/testQueue</value>

</property>

<property name="jndiTemplate">

<ref local="jndiTemplate"/>

</property>

</bean>

<bean id="jndiTemplate" class="org.springframework.jndi.JndiTemplate">

<property name="environment">

<props>

<prop key="java.naming.factory.initial">org.jnp.interfaces.NamingContextFactory</prop>

<prop key="java.naming.provider.url">jnp://localhost:1099</prop>

<prop key="java.naming.factory.url.pkgs">org.jboss.naming:org.jnp.interfaces</prop>

</props>

</property>

</bean>

（2）JndiObjectTargetSource

使用JndiObjectTargetSource，例如

<bean id="queueTarget" class="org.springframework.jndi.JndiObjectTargetSource">

<property name="jndiName">

<value>queue/testQueue</value>

</property>

</bean>

（3）JndiObjectFactoryBean：

使用JndiObjectFactoryBean，例如

<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName">

<value>java:/MySqlDS</value>

</property>

</bean>

## 总结（事例）：

### 数据源的配置：

（1）配置可以访问到同一应用服务器的jndi数据源

<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName">

<value>jdbc/cqccms</value>

</property>

</bean>

（2）配置能访问远程jndi数据源

<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName">

<value>jdbc/cqccms</value>

</property>

<property name="jndiEnvironment">

<props>

<prop key="java.naming.factory.initial">

weblogic.jndi.WLInitialContextFactory

</prop>

<prop key="java.naming.provider.url">t3://172.16.101.42:7001</prop>

<prop key="java.naming.security.principal">weblogic</prop>

<prop key="java.naming.security.credentials">weblogic</prop>

</props>

</property>

</bean>

### 事务的配置：

不过事务对象远程访问好像没那么顺利，同理查看org.springframework.transaction.jta.JtaTransactionManager类，没有发现上述方法，不过看到了setJndiTemplate()方法，也不错现配置一个jndiTemplate,如下：

<bean id="jndiTemplate" class="org.springframework.jndi.JndiTemplate" singleton="true"

lazy-init="default" autowire="default" dependency-check="default">

<property name="environment">

<props>

<prop key="java.naming.factory.initial">

weblogic.jndi.WLInitialContextFactory

</prop>

<prop key="java.naming.provider.url">t3://172.16.101.42:7001</prop>

<prop key="java.naming.security.principal">weblogic</prop>

<prop key="java.naming.security.credentials">weblogic</prop>

</props>

</property>

</bean>

然后在配置一下transactionManager,如下

<bean id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager" singleton="true"

lazy-init="default" autowire="default" dependency-check="default">

<property name="jndiTemplate">

<ref local="jndiTemplate" />

</property>

<property name="userTransactionName">

<value>weblogic/transaction/UserTransaction</value>

</property>

</bean>

不过JtaTransactionManager默认找jndi name为UserTransaction的jta对象，在同一应用服务器上可以，远程访问不到，后直接查看weblogic的jndi树找到这个jndi名weblogic/transaction/UserTransaction，配置上测试果然成功。

Xml代码

1.<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

2. <property name="jndiName">

3. <value>java:/comp/env/jdbc/test</value>

4. </property>

5.</bean>

<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean"> <property name="jndiName"> <value>java:/comp/env/jdbc/test</value> </property> </bean>

jndi-lookup:

Xml代码

1.<jee:jndi-lookup id="simple" jndi-name="java:/comp/env/jdbc/test">

2.</jee:jndi-lookup>

<jee:jndi-lookup id="simple" jndi-name="java:/comp/env/jdbc/test"> </jee:jndi-lookup>

再贴一下tomcat连接池的配置方法：

tomcat连接池的配置是非常简单的，5.5以上的配置，在不改变tomcat的任何配置的情况下建立连接池：

1 在项目的WebRoot下建立META-INF文件夹

2 在META-INF下建立context.xml文件

3 在context.xml中加入如下样例代码

XML code<Context path="/appName" docBase="appName"

debug="5" crossContext="true" reloadable="false"

cachingAllowed="true" cacheMaxSize="20480"

cacheTTL="10000">

<Resource name="jdbc/mysql" auth="Container" removeAbandoned="true"

removeAbandonedTimeout="60" logAbandoned="true"

type="javax.sql.DataSource" maxActive="50" maxIdle="10"

maxWait="10000" username="root" password="root"

driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/dbname?autoReconnect=true" />

</Context>

在web.xml中加入

XML code<resource-ref>

<description>DB Connection</description>

<res-ref-name>jdbc/mysql</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

以上配置文件，参数自己在重新设置设置，别忘了把驱动放到你的项目的classpath下

再加一点内容：

如果用hibernate，那么在hibernate.cfg.xml中这么配置

XML code<hibernate-configuration>

<session-factory name="java:hibernate/SessionFactory">

<!-- JNDI datasource -->

<property name="connection.datasource">

java:comp/env/jdbc/mysql

</property>

<property name="dialect">

org.hibernate.dialect.MySQLInnoDBDialect

</property>

</hibernate-configuration>