# 1、概念

  JNDI是Java Naming and Directory Interface（JAVA命名和目录接口）的英文简写，它是为JAVA应用程序提供命名和目录访问服务的API（Application Programing Interface，应用程序编程接口）。

JNDI中的命名（Naming），就是将Java对象以某个名称的形式绑定（binding）到一个容器环境（Context）中，以后调用容器环境（Context）的查找（lookup）方法又可以查找出某个名称所绑定的Java对象。读者也许会感到奇怪：自己创建一个Java对象，将其绑定到JNDI容器环境中后又查询出来，这有什么意思？在真实的项目应用中，通常是由系统程序或框架程序先将资源对象绑定到JNDI环境中，以后在该系统或框架中运行的模块程序就可以从JNDI环境中查找这些资源对象了。例如，Tomcat服务器在启动时可以创建一个连接到某种数据库系统的数据源（DataSource）对象，并将该数据源（DataSource）对象绑定到JNDI环境中，以后在这个Tomcat服务器中运行的Servlet和JSP程序就可以从JNDI环境中查询出这个数据源（DataSource）对象进行使用，而不用关心数据源（DataSource）对象是如何创建出来的，这种方式极大地增强了系统的可维护性，当数据库系统的连接参数发生变更时，这只是Tomcat系统管理员一个人要关心的事情，而与所有的应用程序开发人员无关。

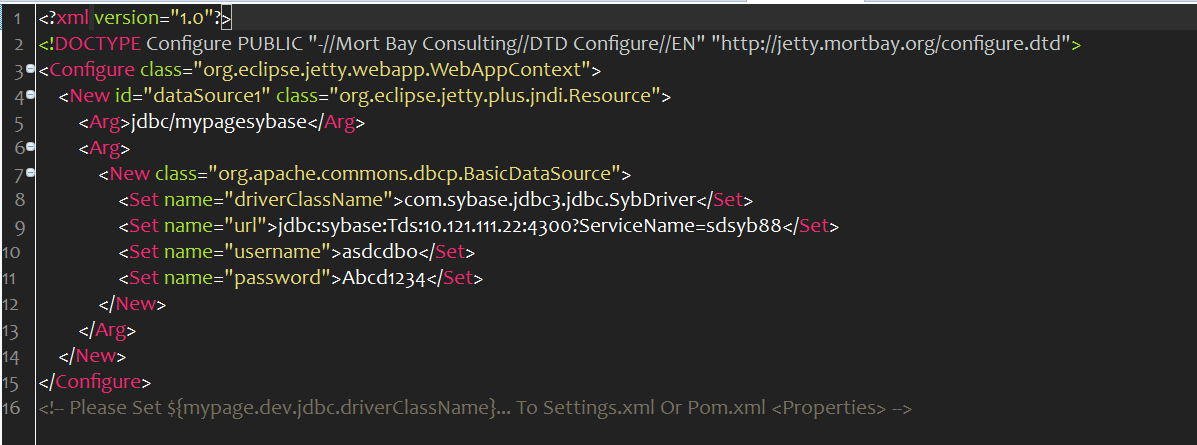
在JNDI中不存在着“根”Context的概念，也就是说，执行JNDI操作不是从一个“根”Context对象开始，而是可以从Context树中的任意一个Context开始。无论如何，程序必须获得一个作为操作入口的Context对象后才能执行各种JNDI命名操作，为此，JNDI API中提供了一个InitialContext类来创建用作JNDI命名操作的入口Context对象。Context是一个接口，Context对象实际上是Context的某个实现类的实例对象，选择这个具体的Context实现类并创建其实例对象的过程是由一个Context工厂类来完成的，这个工厂类的类名可以通过JNDI的环境属性java.naming.factory.initial指定，也可以根据Context的操作方法的url参数的Schema来选择。

https://blog.csdn.net/shunzi1046/article/details/62227733

# 2、代码示例

## （1）在jetty-env.xml文件中配置数据源

代码如下：



## （2）在pom.xml文件中配置了sybase驱动的dependency

<dependency>

<groupId>org.apache</groupId>

<artifactId>jdbc3</artifactId>

<version>1.0</version>

<scope>system</scope>

<systemPath>${basedir}/src/main/webapp/WEB-INF/lib/jconn3.jar</systemPath>

</dependency>

## （3）在WEB-INF/lib目录下数据库驱动jar包

## （4）在web.xml文件中导入数据源

<resource-ref>

<description>mypage</description>

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

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

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

</resource-ref>

说明：数据源的名称一定要与Jetty-env.xml文件中的数据源名称相同。

## （5）在dao层的实现类编写代码

Connection conn = null;

Context context;

DataSource ds = null;

try {

try {

context = new InitialContext();

ds = (DataSource) context.lookup(“数据源的名称”);

} catch (NamingException e) {

e.printStackTrace();

}

conn = ds.getConnection();

说明：重点三个对象：Connection、Context、DataSource。建立连接之后就可以查询数据库了。