JNDI & Connection pooling

Objectives

At the end of this session, you will be able to,

- Describe architecture of JNDI
- Describe Connection Pool
- Demonstrate implementation of DataSource

Agenda

Following points to be covered in the session,

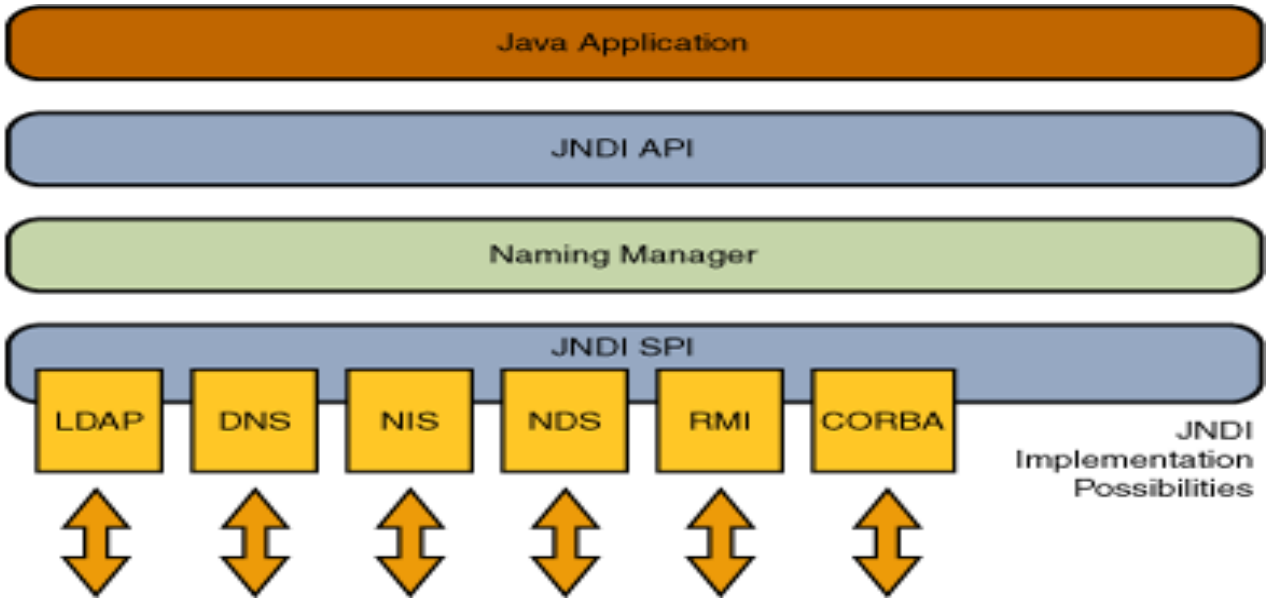
- JNDI Architecture
- Connection Pool
- Data Source implementation in JEE

Overview of JNDI

- The **Java Naming and Directory Interface**™ (JNDI) is an application programming interface (API) that provides naming and directory functionality to applications written using the Java™ programming language.
- It is defined to be independent of any specific directory service implementation. Thus a variety of directories -new, emerging, and already deployed can be accessed in a common way.
- **A naming service maps developer friendly names to objects . So that the applications can access the same object with the bound name.**

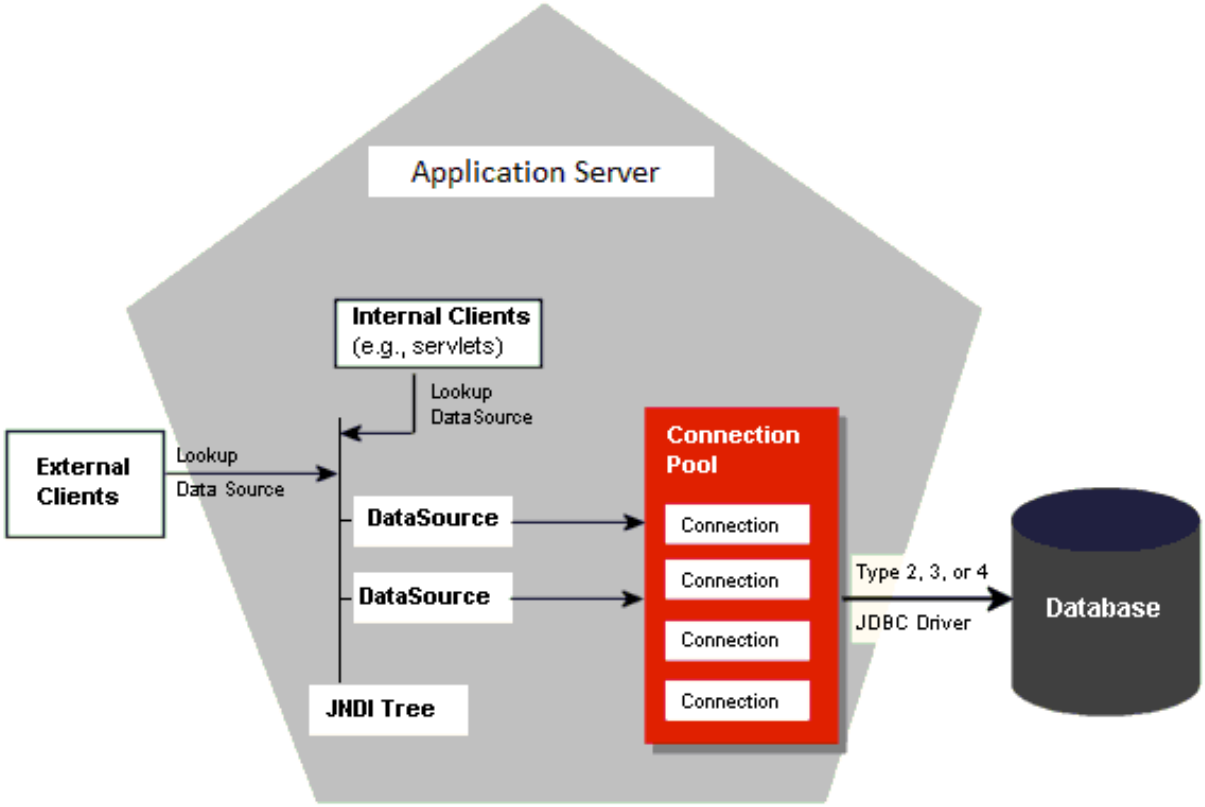
JNDI Architecture

- The JNDI architecture consists of an API and a service provider interface (SPI). Java applications use the JNDI API to access a variety of naming and directory services.
- The SPI enables a variety of naming and directory services to be plugged in transparently, thereby allowing the Java application using the JNDI API to access their services. See the following figure:



Connection Pool

- As we have seen importance and need of connection pool in JDBC , lets see how we can implement it in server.



Connection Pool (Cont..)

- A connection pool operates by performing the work of creating connections ahead of time, In the case of a JDBC connection pool, a pool of **Connection** objects is created at the time the application server (or some other server) starts.
- These objects are then managed by a **pool manager** that disperses connections as they are requested by clients and returns them to the pool when it determines the client is finished with the **Connection** object.

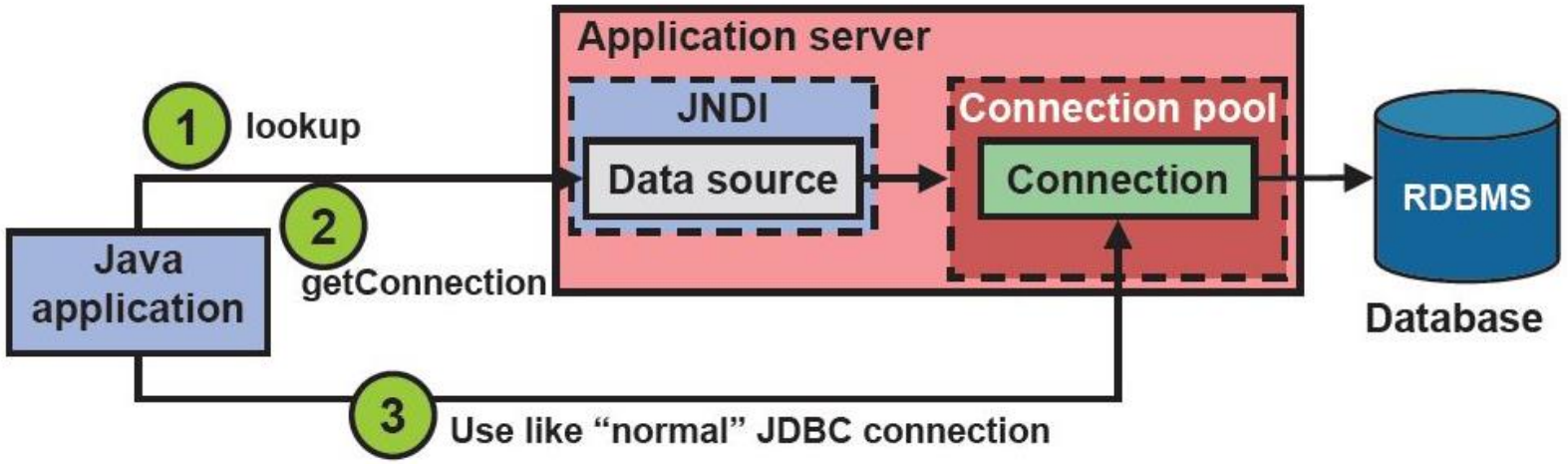
- When the connection pool server starts, it creates a predetermined number of **Connection** objects. A client application would then perform a JNDI lookup to retrieve a reference to a **DataSource** object that implements the **ConnectionPoolDataSource** interface. The client application would not need make any special provisions to use the pooled data source; the code would be no different from code written for a nonpooled **DataSource.**

Connection Pool (Cont..)

- When the client application requests a connection from the **ConnetionPoolDataSource,** the data source implementation would retrieve a physical connection to the client application. The **ConnectionPoolDataSource** would return a **Connection** object that implemented the **PooledConnection** interface.

Data Source

- When a request comes towards server the following way is used to process it.
- Following diagram also shows relation between Data Source, JNDI and Connection Pool.



Data Source coding implementation

- To implement JNDI and get a connection object from connection pool we need to write down following code in servlet instead of normal JDBC code

This is JNDI name for lookup where config. details are stored.

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

- For using these APIs, do import javax.naming.InitialContext and javax.sql.DataSource in java code

Summary

In this session, we have covered,

- JNDI
- Connection Pool
- Data Source Configuration

Thank You

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.