Oracle Article

https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

https://www.baeldung.com/java-jdbc
https://jenkov.com/tutorials/jdbc/index.html

https://docs.oracle.com/javase/tutorial/jdbc/basics/sqldatasources.html

Java Naming and Directory Interface (JNDI) is a Java API that provides naming and directory functionality to Java applications. The JNDI service allows Java applications to access and manipulate objects stored in naming and directory services, such as LDAP (Lightweight Directory Access Protocol) servers, DNS (Domain Name System), and RMI (Remote Method Invocation) registries.

The primary use cases of JNDI service include:

1. **Naming and Lookup**: JNDI allows Java applications to look up and access objects stored in a naming service. These objects can include database connections, JMS (Java Message Service) resources, and other resources managed by a Java EE application server.

2. **Resource Injection**: In Java EE applications, JNDI is commonly used for resource injection. For example, a servlet or enterprise bean can obtain a database connection or other resources by looking up the resource using JNDI, rather than hard-coding the resource configuration in the application code.

3. **Directory Services**: JNDI can be used to access and manipulate directory services, such as LDAP directories. This allows Java applications to perform operations like searching, adding, modifying, and deleting entries in LDAP directories.

4. **Integration with Legacy Systems**: JNDI provides a standardized way for Java applications to integrate with legacy systems and other non-Java environments that support naming and directory services.

Overall, the JNDI service provides a flexible and extensible mechanism for Java applications to interact with naming and directory services, making it easier to build distributed and scalable Java applications.

https://www.baeldung.com/java-connection-pooling

https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

https://www.baeldung.com/jndi

The provided code snippets demonstrate how to use Spring's `SimpleNamingContextBuilder` and `JndiTemplate` to set up and access a JNDI (Java Naming and Directory Interface) context within a Spring application. Let's break down the purpose of each component:

1. **SimpleNamingContextBuilder**:
   - **Purpose**: `SimpleNamingContextBuilder` is a helper class provided by Spring to create and configure a JNDI naming context.
   - **Usage**: It allows you to define and register objects (resources, beans, etc.) within the JNDI context using a simple API. This can be useful for testing or standalone applications where you need to simulate a JNDI environment.

2. **JndiTemplate**:
   - **Purpose**: `JndiTemplate` is a helper class provided by Spring for performing JNDI operations.
   - **Usage**: It abstracts away the complexity of JNDI operations and provides a simplified API for accessing and manipulating objects in a JNDI context. In the provided example, `JndiTemplate` is used to obtain the `InitialContext`, which represents the starting point for accessing the JNDI naming and directory service.

Overall, these components allow you to work with JNDI in a more convenient and streamlined manner within a Spring application. They simplify the setup and usage of JNDI resources, making it easier to integrate with JNDI-based systems and services.

https://stackoverflow.com/questions/4365621/what-is-jndi-what-is-its-basic-use-when-is-it-used