Welcome to JDBC tutorial. **Java DataBase Connectivity** (JDBC) is one of the most widely used API in enterprise applications. This is because most of the applications use some sort of database connectivity. I have recently posted a lot of JDBC tutorials related to basic JDBC, DataSource and it’s integration with Spring Framework.

This is a index post for all the jdbc tutorials I had posted earlier. If you are new to JDBC then you should go through these JDBC tutorials in order for better understanding.

1. [**JDBC Example**](https://www.journaldev.com/2471/jdbc-example-mysql-oracle)

JDBC API helps us in writing loosely coupled code from database drivers. This article explains about different types of database drivers we have and which one is the most commonly used and why. This is a good post to start with to learn about Database Connection, Statements and ResultSet. How they all work together to perform specific database operations.

This article also shows you how to write JDBC programs in order to keep them loosely coupled from Database drivers, that helps in switching from one database server to another easily with just configuration changes.

1. [**JDBC Statement vs PreparedStatement**](https://www.journaldev.com/2489/jdbc-statement-vs-preparedstatement-sql-injection-example)

JDBC API provides two ways to communicate with database – Statement and PreparedStatement. Statement is easy to use but it can lead to SQL injection, that is very common way of hacking any application. This article clearly shows how SQL injection can be performed with SQL Statements and why we should use PreparedStatement to avoid SQL injection attacks.

This article goes further in explaining some of the major benefits we get from using PreparedStatement over Statement such as caching, object oriented programming and elegant looking code.

1. [**JDBC PreparedStatement IN Clause Alternatives**](https://www.journaldev.com/2521/java-preparedstatement-in-clause-alternatives)

Since JDBC PreparedStatement is pre-compiled, we can’t use it with IN clause. Rather than going back to Statement, there are some alternative approaches that we can use to get over this shortcomings of Prepared Statement. This article provides four different alternative approaches that we can take to support IN clause with prepared statements. You should read it because you never know when you are going to need it, also it’s one of the most asked interview question related to JDBC.

1. [**JDBC Batch Processing**](https://www.journaldev.com/2494/jdbc-batch-insert-update-mysql-oracle)

If you are working with bulk data and have to execute a lot of queries, it’s not good idea to do it one by one. JDBC provides batch processing support that will be much more faster than processing single query at a time. This article shows you how to write programs for batch processing.

It goes further into discussing what to do if there is an exception thrown by one of the queries in the batch.

1. [**JDBC CallableStatement Example**](https://www.journaldev.com/2502/callablestatement-in-java-example)

We can use JDBC API CallableStatement to execute stored procedures. Oracle Database provides Cursors and DB Objects that we can use in stored procedures IN/OUT parameters. This tutorial provides specific details of these with example programs.

1. [**JDBC DataSource Example**](https://www.journaldev.com/2509/java-datasource-jdbc-datasource-example)

Most of the times we are looking for more than a database connection. Creating a connection is a heavy process and it’s not a good idea to let every part of the program to create it’s own connection. This can lead to resource starvation and slow performance.

That’s why we use Connection Pooling in most of the enterprise application. Most of the database drivers provide DataSource implementation classes that can be used in the connection pool.

This tutorial provide example of MySQL and Oracle DataSource and how to use them. The article also provide details about Apache DBCP that works as a wrapper around the different DataSource implementations to achieve loose coupling.

1. [**JDBC Transaction Management**](https://www.journaldev.com/2483/java-jdbc-transaction-management-savepoint)

Transaction Management is important when we have a group of queries to execute and we want to make sure either all of them executes or none of them. We can set the connection auto commit to false, to achieve transaction management. If everything goes fine, we can commit the transaction or if there are any exceptions we can rollback the whole transaction.

It also explains about the Savepoint that we can use to rollback to a particular point in the transaction. You can think of Savepoint as milestones in the transaction.

1. [**Tomcat JNDI DataSource Example**](https://www.journaldev.com/2513/tomcat-datasource-jndi-example-java)

Most of the servlet containers supports JNDI resources for DataSource that we can use to offload the transaction management and connection pooling tasks to the container. This article explains about different ways through which we can configure DataSource in Apache Tomcat server and use JNDI context lookup to get the DataSource and work with it.

1. [**Spring JDBC and JdbcTemplate Example**](https://www.journaldev.com/2593/spring-jdbc-example)

This article provides details of JDBC integration with Spring Framework. We can either chose to use standard JDBC API or get the benefits of Spring JdbcTemplate that helps us in removing all the boiler plate code that comes with JDBC, for example open/close connection, statement, result sets etc.

1. [**Spring Transaction Management**](https://www.journaldev.com/2603/spring-transaction-management-jdbc-example)

Spring provides built-in support for transaction management. Transaction management is a cross-cutting concerns and Spring AOP approach with declarative transaction management is very simple and elegant to use. A sample project explaining different aspects of transaction management in Spring Framework for JDBC operations.

1. [**Spring DataSource JNDI Example**](https://www.journaldev.com/2597/spring-datasource-jndi-with-tomcat-example)

Spring framework is very popular because it provides configuration based solution for most of the common tasks. That’s why spring framework provides simple configurations to lookup JNDI context and get DataSource defined by the servlet container. A tutorial with Spring MVC Example Project to show you how to do it easily.

1. [**JSF JDBC Integration Example**](https://www.journaldev.com/7068/jsf-database-example-mysql-jdbc)

JSF is one of the most widely used component based framework. This tutorial explains how we can integrate JDBC API with JSF framework.

1. [**JDBC Interview Questions and Answers**](https://www.journaldev.com/2529/jdbc-interview-questions-and-answers)

A collection of 40 JDBC related interview questions with detailed answers to help you in Java interviews to conquer different kinds of JDBC questions.

I will be adding more posts in JDBC tutorial, so please bookmark the post for future reference too.