SpringJDBC(SpringDAO)

=> It provides abstraction on plain jdbc technology and simplifies jdbc style persistence logic development
by avoiding boiler plate code.

PlainJDBC

a. load the driver
b. establish the connection
c. create JDBC Statement object

d. send and exeucte query
e. gather results and process results

f. perform exception handling
g. perform transaction management

h. close jdbc objects

In the above steps d,e specifies application logic whereas remaining steps corresponds to boiler plate code.

=> In case of SpringJDBC application the boiler plate is take care by SpringJDBC API's.

a. Inject JdbcTemplate class having DataSource object.
[Steps :: a,b,c,f,g,h] boiler plate code will be taken care by JdbcTemplate
b. Send and execute the query
c. Gather results and process results
[Application specific logic should be taken care by programmers]

SpringJDBC Advantages

1. Supports both positional(?) and named argument(:=name)
2. we can get "select query" results in different format directly with the support of
query()queryxXXX(),queryForList(),queryForMap(),queryForObject()....
3. customization of Result is bit easy because it uses "CallBackInterface".
4. Provides abstration on plain jdbc and avoids boiler plate code(common logic will be generated automatically)
5. Give detailed exception class hierarchy which is called "DataAccessException"class hierarchy
a. Exceptions are made as unchecked excpetion.
b. Excpetion handling is optional.
c. Supports exception propogation by default.
d. These are exceptions which are common for SpringORM,SpringDataJPA modules also.
e. Spring JDBC internally uses Exception rethrowing concept to convert all the checked exceptions to uncheckedExceptions.
6. Simplifies the call of StoredProcedure
7. Gives the great support to work with Generics and var-args
8. It can generate insert sql query dynamically based on the given dbtablename,colname,and colvalues.

Different approaches of developing persistence logic

==========

a. using JdbcTemplate
b. using NamedParameterJdbcTemplate
c. Using SimpleJdbcInsert,SimpleJdbcCall
d. MappingSQLOperation as subclass

Simple Jdbc call to work with the callable statements

JdbcTemplate with CallBackInterfaces
==========

1.RowMapper<T>
2.ResultSetExtractor
3.RowCallBackHandler