# Data access with JDBC

## Introduction to Spring Framework JDBC

The value-add provided by the Spring Framework JDBC abstraction is perhaps best shown by the sequence of actions outlined in the table below. The table shows what actions Spring will take care of and which actions are the responsibility of you, the application developer.



The Spring Framework takes care of all the low-level details that can make JDBC such a tedious API to develop with.

### Choosing an approach for JDBC database access

You can choose among several approaches to form the basis for your JDBC database access. In addition to three flavors of the JdbcTemplate, a new **SimpleJdbcInsert** and **SimplejdbcCall** approach optimizes database metadata, and the RDBMS Object style takes a more object-oriented approach similar to that of JDO Query design. Once you start using one of these approaches, you can still mix and match to include a feature from a different approach. All approaches require a JDBC 2.0-compliant driver, and some advanced features require a JDBC 3.0 driver.

* **JdbcTemplate** is the classic Spring JDBC approach and the most popular. This "lowest level" approach and all others use a JdbcTemplate under the covers.
* **NamedParameterJdbcTemplate** wraps a JdbcTemplate to provide named parameters instead of the traditional JDBC "?" placeholders. This approach provides better documentation and ease of use when you have multiple parameters for an SQL statement.
* **SimpleJdbcInsert** and **SimpleJdbcCall** optimize database metadata to limit the amount of necessary configuration. This approach simplifies coding so that you only need to provide the name of the table or procedure and provide a map of parameters matching the column names. This only works if the database provides adequate metadata. If the database doesn’t provide this metadata, you will have to provide explicit configuration of the parameters.
* **RDBMS Objects** including **MappingSqlQuery**, **SqlUpdate** and **StoredProcedure** requires you to create reusable and thread-safe objects during initialization of your data access layer. This approach is modeled after JDO Query wherein you define your query string, declare parameters, and compile the query. Once you do that, execute methods can be called multiple times with various parameter values passed in.

### Package hierarchy

The Spring Framework’s JDBC abstraction framework consists of four different packages, namely core, datasource, object, and support.

org.springframework.jdbc.core.JdbcTemplate

org.springframework.jdbc.core.simple.SimpleJdbcInsert and SimpleJdbcCall

org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate

org.springframework.jdbc.datasource.DataSource

org.springfamework.jdbc.datasource.embedded.\* -->Java database engines such as HSQL, H2, and Derby

org.springframework.jdbc.object.\* --> classes that represent RDBMS queries, updates, and stored procedures as thread-safe, reusable objects.

org.springframework.jdbc.support.\* --> provides SQLException translation functionality and some utility classes.

org.springframework.dao.\* --> Exceptions thrown during JDBC processing translated to this package classes

## Using the JDBC core classes to control basic JDBC processing and error handling

## <https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/data-access.html#jdbc-connections>