Spring Jdbc

- If you want to connect with database, one of the best way is JDBC
- Spring JDBC dependent on DataSource.
- Data Source is nothing but a URL, Driver Class, UserName, Password, we can provide all these in application.properties file.

Ex:--
```
@Autowired
private JdbcTemplate jdbcTemplate;
```

- Here we have lot of methods available for JdbcTemplate class, based on the requirement we can use all these methods.

Ex:--
```
public List<User> findAll(){
      return jdbcTemplate.query("select * from logindetails", new
                      BeanPropertyRowMapper(User.class));
}
User.class should match the return of query properties.
```

Note:-- if you are connecting with Oracle, we should add the oracle dependency. Before adding the dependency by default it is not available, we have to set the oracle(ojdbc driver) to our local maven repository.

Application.properties ::

```
Spring.datasource.driver-class-name=
spring.datasource.url=
spring.datasource.username=
spring.datasource.password=
```

```
public Customer findByCustomerId3(Long id) {

        String sql = "SELECT * FROM CUSTOMER WHERE ID = ?";

        return jdbcTemplate.queryForObject(sql, new Object[]{id}, (rs, rowNum) ->
                new Customer(
                        rs.getLong("id"),
                        rs.getString("name"),
                        rs.getInt("age"),
                        rs.getTimestamp("created_date").toLocalDateTime()
                ));

    }
```
Methods:

- update(); --- delete operation or update operation or insert operation ; returns int…
- Query();   /// select or retrieve operations
- queryForObject();
- queryForList();
- queryForMap();

update(); insert/ update/delete

below example **"sql"** is query where we can create as a prepared statement query, and **"`new Object[]{id}`"** is a parameter how many ? have in your query that many we can pass it from here.

Ex:-- jdbcTemplate.update(sql, `new Object[]{id}; //returns int..`

RowMapper:-

- most of the times whenever we are using Spring JDBC, we have to use the RowMapper concept / ResultSetExtractor..
- RowMapper is an interface having one method call mapRow();

Ex:-Class UserRowMapper implements RowMapper<User>{

```
@override
public User mapRow(ResultSet rs, int rowNum){

    User user = new User();

    User.setId(rs.getString("id"));
    // //  /// ----
 }
}
```

```
public List<User> findAll(){
                        return jdbcTemplate.query("select * from logindetails", new
                                        new UserRowmapper());
            }
```

Spring

- We have to create beans manually for datasource and jdbc template

```
@Configuration
public class SpringJDBCConfiguration {

@Bean
  public DataSource dataSource() {

     DriverManagerDataSource dataSource = new DriverManagerDataSource();

     dataSource.setDriverClassName("com.mysql.jdbc.Driver");
     dataSource.setUrl("jdbc:mysql://localhost:3306/TestDB");//change url
     dataSource.setUsername("userid");//change userid
     dataSource.setPassword("password");//change pwd
     return dataSource;
  }

  @Bean
  public JdbcTemplate jdbcTemplate() {
     JdbcTemplate jdbcTemplate = new JdbcTemplate();
     jdbcTemplate.setDataSource(dataSource());
     return jdbcTemplate;
  }
}
```

Hibernate(Spring)

- HibernateTemplate on sessionFactory and sessionFactory dependent on session but session dependent on Datasource.

JPA(Spring)

- Jpa dependent on entityManagerFactory this will dependent on EntityManager but this is dependent on Datasource