Spring Jdbc Hibernate Template

Presented by

Sagar
Java Consultant

spring

Spring and JDBC Template

It internally uses JDBC API, but eliminates problems of JDBC API.

Scenario : Pure JDBC

- Creating connection, statement, resultset objects and closing them.

- We need to handle the transactions - commit() / rollback()

Why Spring JDBC Template ?

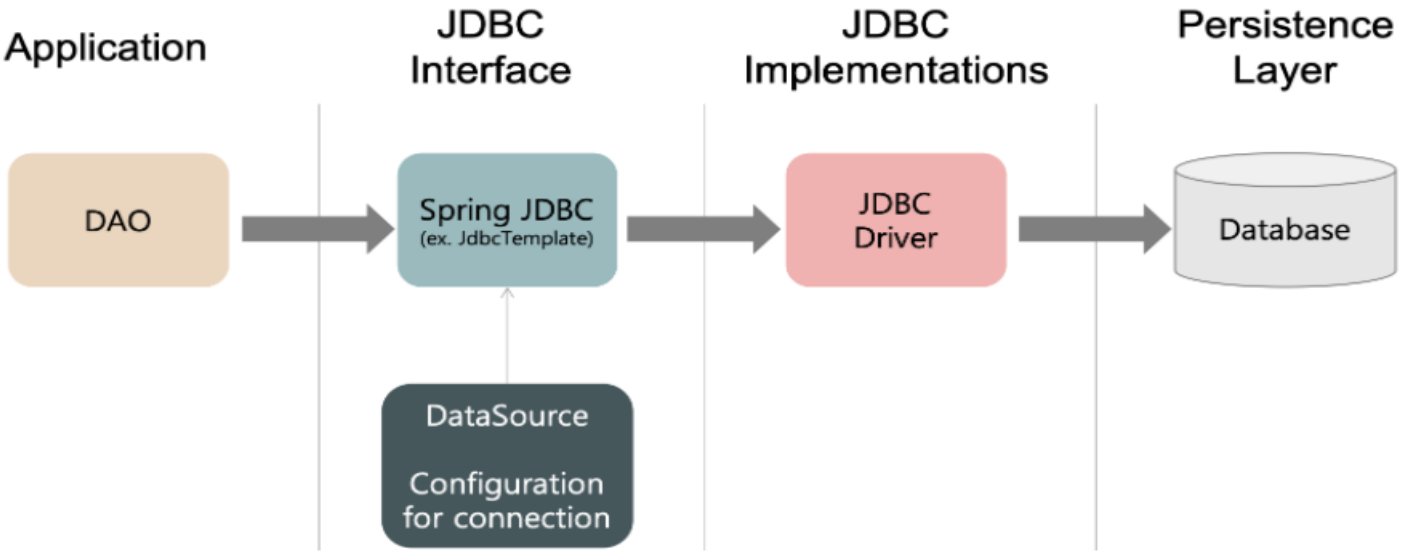
- Spring JdbcTemplate eliminates certain problems of JDBC API
- Provides methods to write the queries directly

Spring JDBC Template ..

•It is the central class in the Spring JDBC support classes.

•It takes care of creation and release of resources such as creating and closing of connection object etc.

•So it will not lead to any problem if you forget to close the connection.



spring

Spring JDBC API

Spring JDBC Template API :

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.datasource.SimpleDriverDataSource;

static JdbcTemplate jdbcTemplateObj;
static SimpleDriverDataSource dataSourceObj;

// Database Configuration Parameters
static String DB_USERNAME = "root";
static String DB_PASSWORD = "root";
static String DB_URL = "jdbc:mysql://localhost:3306/springjdbcdb";
```

spring

Spring JDBC API …

```
public static SimpleDriverDataSource getDatabaseConnection() {
    dataSourceObj = new SimpleDriverDataSource();
    try {
        dataSourceObj.setDriver(new com.mysql.cj.jdbc.Driver());
        dataSourceObj.setUrl(DB_URL);
        dataSourceObj.setUsername(DB_USERNAME);
        dataSourceObj.setPassword(DB_PASSWORD);
    } catch (SQLException sqlException) {
        sqlException.printStackTrace();
    }
    return dataSourceObj;
}
```

** Example

CRUD Methods

JDBC Template Object :

```
jdbcTemplateObj = new JdbcTemplate(getDatabaseConnection());
```

To insert data :

```
jdbcTemplateObj.update(sqlInsertQuery, "Sagar", "sagar@mail.com", "India", "954786321");
System.out.println("Row Inserted!");
```

To retrieve data :

RowMapper interface is used to fetch the records from the database using the **query()** method of the **JdbcTemplate** class.

```
String sqlSelectQuery = "SELECT name, email, address, telephone FROM Customer";
List<Customer> customerList = jdbcTemplateObj.query(sqlSelectQuery, new RowMapper()
```

CRUD Methods …

To update data:

```
String sqlUpdateQuery = "UPDATE Customer set email=? where name=?";
jdbcTemplateObj.update(sqlUpdateQuery, "administrator101@gmail.com", "Editor 101");
```

To delete data:

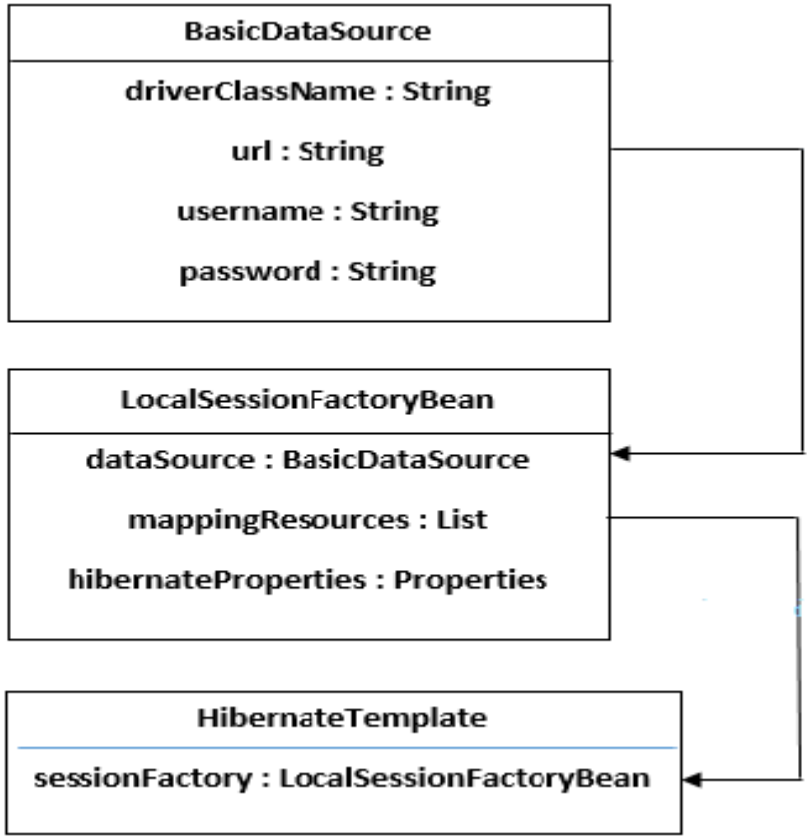
```
String sqlDeleteQuery = "DELETE FROM Customer where name=?";
jdbcTemplateObj.update(sqlDeleteQuery, "Editor 104");
```

Spring Hibernate Template

- In hibernate framework, we provide all the database information **hibernate.cfg.xml** file.
- But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file.
- We can provide all the information in the **applicationContext.xml** file.
- It contains information of DataSource, SessionFactory etc.
- Don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc.

Spring Hibernate Template Architecture

Spring Hibernate Architecture



Spring Hibernate Template Configuration

```
<tx:annotation-driven/>
<bean class="org.springframework.jdbc.datasource.DriverManagerDataSource" id="ds">
  <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
  <property name="url" value="jdbc:mysql://localhost:3306/springhibernatedb"/>
  <property name="username" value="root"/>
  <property name="password" value="root"/>
</bean>
<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
  <property name="dataSource" ref="ds"/>
▼<property name="hibernateProperties">
  ▼<props>
     <prop key="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop>
     <prop key="hibernate.show_sql">true</prop>
     <!--  <prop key="hibernate.format_sql">true</prop>  -->
     <prop key="hibernate.hbm2ddl.auto">update</prop>
   </props>
  </property>
```

Spring Hibernate Template methods

```
import org.springframework.orm.hibernate5.HibernateTemplate;
        private HibernateTemplate hTemplate;
        ……….
        hTemplate.save(student);
        ……….

        hTemplate.get(Student.class,id);
        ……….
        hTemplate.loadAll(Student.class);
        ……….
        hTemplate.update(student);
        ……….
        hTemplate.delete(student);
```

**Demo

Thank You