**Basic Configuration:**

Explain how to configure Spring JDBC with JdbcTemplate in a Spring application. Provide an example of setting up data source configuration (DataSource bean) and configuring JdbcTemplate bean in XML configuration (applicationContext.xml).

Configuring Spring JDBC with JdbcTemplate involves setting up a DataSource bean and a JdbcTemplate bean in the Spring application context configuration file (applicationContext.xml). Here's a step-by-step guide along with an example:

**Step 1: Define DataSource Bean**

The DataSource bean in Spring defines the database connection properties and is used by the JdbcTemplate for database operations.

**Example of DataSource configuration in applicationContext.xml:**

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/exampledb" />

<property name="username" value="username" />

<property name="password" value="password" />

</bean>

Replace com.mysql.cj.jdbc.Driver with the appropriate JDBC driver class for your database. Modify the url, username, and password properties according to your database configuration.

### Step 2: Define JdbcTemplate Bean

The JdbcTemplate bean encapsulates the DataSource and simplifies the use of JDBC operations.

#### Example of JdbcTemplate configuration in applicationContext.xml:

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">

<property name="dataSource" ref="dataSource" />

</bean>

In this example:

* The dataSource property of the JdbcTemplate bean is set to reference the dataSource bean defined earlier. This links the JdbcTemplate to the DataSource.

### Step 3: Using JdbcTemplate in Spring Beans

Once configured, you can inject the jdbcTemplate bean into your Spring beans (like DAO classes) for performing database operations.

#### Example of using JdbcTemplate in a DAO class:

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

@Repository

public class MyDao {

private JdbcTemplate jdbcTemplate;

@Autowired

public MyDao(JdbcTemplate jdbcTemplate) {

this.jdbcTemplate = jdbcTemplate;

}

public void insertData(String data) {

String sql = "INSERT INTO my\_table (data\_column) VALUES (?)";

jdbcTemplate.update(sql, data);

}

// Other methods for data retrieval, update, delete, etc.

}

In this example:

* The jdbcTemplate bean is injected into the MyDao class constructor using @Autowired.
* The insertData method demonstrates using the JdbcTemplate to perform an insert operation into the database.