Spring JDBC Tutorial

- In this example you will learn how the Spring *JDBCTemplate* simplifies the code you need to write to perform the database-related operations. The *insertForum()* method below shows the amount of code you need to write to insert data using JDBC.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.sql.DataSource;

import com.vaannila.domain.Forum;

public class JDBCForumDAOImpl implements ForumDAO {

        private DataSource dataSource;

        public void setDataSource(DataSource dataSource) {
                this.dataSource = dataSource;
        }
public void insertForum(Forum forum) {
                /**
                 * Specify the statement
                 */
                String query = "INSERT INTO FORUMS (FORUM_ID, FORUM_NAME,
FORUM_DESC) VALUES (?,?,?)";
                /**
                 * Define the connection and preparedStatement parameters
                 */
                Connection connection = null;
                PreparedStatement preparedStatement = null;
                try {
                        /**
                         * Open the connection
                         */
                        connection = dataSource.getConnection();
                        /**
                         * Prepare the statement
                         */
                        preparedStatement = connection.prepareStatement(query);
                        /**
                         * Bind the parameters to the PreparedStatement
                         */
                        preparedStatement.setInt(1, forum.getForumId());
                        preparedStatement.setString(2, forum.getForumName());
                        preparedStatement.setString(3, forum.getForumDesc());
                        /**
```

```
                         * Execute the statement
                         */
                        preparedStatement.execute();
                } catch (SQLException e) {
                        /**
                         * Handle any exception
                         */
                        e.printStackTrace();
                } finally {
                        try {
                                /**
                                 * Close the preparedStatement
                                 */
                                if (preparedStatement != null) {
                                        preparedStatement.close();
                                }
                                /**
                                 * Close the connection
                                 */
                                if (connection != null) {
                                        connection.close();
                                }
                        } catch (SQLException e) {
                                /**
                                 * Handle any exception
                                 */
                                e.printStackTrace();
                        }
                }

        }

}
```

- As you can see they are mostly boilerplate code required to manage the resources and handle exceptions. The code below shows how the Spring *JDBCTemplate* can simplify this task for you.

```
public class ForumDAOImpl implements ForumDAO {

        private JdbcTemplate jdbcTemplate;

        public void setDataSource(DataSource dataSource) {
                this.jdbcTemplate = new JdbcTemplate(dataSource);
        }

        public void insertForum(Forum forum) {
                /**
                 * Specify the statement
                 */
String query = "INSERT INTO FORUMS (FORUM_ID, FORUM_NAME, FORUM_DESC)
VALUES (?,?,?)";
                /**
                 * Specify the values
                 */
                jdbcTemplate.update(query, new Object[] {
Integer.valueOf(forum.getForumId()),
                                forum.getForumName(), forum.getForumDesc()
});
        }
}
```

- Using *JDBCTemplate* you write code only related to inserting the data and all the other boilerplate code are taken care by the template itself. Different *update()* methods are available, you can implement the one that is simple and suites your need. The one we implemented here takes a sql query and an array of *Object* that contains values to be bound to indexed parameters of the query. *JDBCTemplate* is suitable with JDK 1.4 and higher.
- The *selectForum()* method below shows the amount of code you need to write to retrive data using JDBC.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.sql.DataSource;

import com.vaannila.domain.Forum;

public class JDBCForumDAOImpl implements ForumDAO {

        private DataSource dataSource;

        public void setDataSource(DataSource dataSource) {
                this.dataSource = dataSource;
        }
public Forum selectForum(int forumId) {
                /**
```

```
                 * Specify the statement
                 */
                String query = "SELECT * FROM FORUMS WHERE FORUM_ID=?";
                /**
                 * Define the connection, preparedStatement and resultSet
parameters
                 */
                Connection connection = null;
                PreparedStatement preparedStatement = null;
                ResultSet resultSet = null;
                try {
                        /**
                         * Open the connection
                         */
                        connection = dataSource.getConnection();
                        /**
                         * Prepare the statement
                         */
                        preparedStatement = connection.prepareStatement(query);
                        /**
                         * Bind the parameters to the PreparedStatement
                         */
                        preparedStatement.setInt(1, forumId);
                        /**
* Execute the statement
                         */
                        resultSet = preparedStatement.executeQuery();
                        Forum forum = null;
                        /**
                         * Extract data from the result set
                         */
                        if(resultSet.next())
                        {
                                forum = new
Forum(resultSet.getInt("FORUM_ID"), resultSet.getString("FORUM_NAME"),
resultSet.getString("FORUM_DESC"));
                        }
```

```
                return forum;
            } catch (SQLException e) {
                /**
                 * Handle any exception
                 */
                e.printStackTrace();
            } finally {
                try {
                    /**
                     * Close the resultSet
                     */
                    if (resultSet != null) {
                        resultSet.close();
                    }
                    /**
                     * Close the preparedStatement
                     */
                    if (preparedStatement != null) {
                        preparedStatement.close();
                    }
                    /**
                     * Close the connection
                     */
                    if (connection != null) {
                        connection.close();
                    }
                } catch (SQLException e) {
                    /**
                     * Handle any exception
                     */
                    e.printStackTrace();
                }
            }
            return null;
        }
```

Now see how you can remove the boilerplate code using the Spring *JDBCTemplate*.

```
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.sql.DataSource;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;

import com.vaannila.domain.Forum;

public class ForumDAOImpl implements ForumDAO {

        private JdbcTemplate jdbcTemplate;

        public void setDataSource(DataSource dataSource) {
                this.jdbcTemplate = new JdbcTemplate(dataSource);
        }
```

```
public Forum selectForum(int forumId) {
                /**
                 * Specify the statement
                 */
                String query = "SELECT * FROM FORUMS WHERE FORUM_ID=?";
                /**
                 * Implement the RowMapper callback interface
                 */
                return (Forum) jdbcTemplate.queryForObject(query, new
Object[] { Integer.valueOf(forumId) },
                                new RowMapper() {
                                        public Object mapRow(ResultSet
resultSet, int rowNum) throws SQLException {
                                                return new
Forum(resultSet.getInt("FORUM_ID"), resultSet.getString("FORUM_NAME"),

        resultSet.getString("FORUM_DESC"));
                                        }
                                });
        }

}
```

- Here you need to implement the *mapRow()* method of the *RowMapper* callback interface. In the *mapRow()* method, map the single row of the result set to the Forum object. The *queryForObject()* method takes a sql query, an array of *Object* that contains values to be bound to indexed parameters of the query and a *RowMapper* object.
- You need not handle any database-related exceptions explicitly instead Spring JDBC Framework will handle it for you. All the exceptions thrown by the Spring JDBC Framework are subclasses of DataAccessException. The *DataAccessException* is a type of *RuntimeException*, so you are not forced to handle it. The *SQLException* is a checked exception, when you throw the *SQLException* here the Spring JDBC Framework will wrap this checked exception inside one of the subclasses of *DataAccessException* and rethrow it, this eliminates the need to explicitly handle them.

- In the Spring bean configuration file you need to first configure a datasource and then inject it to the DAO class.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

        <bean id="dataSource" destroy-method="close"
class="org.apache.commons.dbcp.BasicDataSource">
<property name="driverClassName" value="org.hsqldb.jdbcDriver"/>
        <property name="url" value="jdbc:hsqldb:hsql://localhost"/>
        <property name="username" value="sa"/>
        <property name="password" value=""/>
    </bean>

        <bean id="forumDAO" class="com.vaannila.dao.ForumDAOImpl">
                <property name="dataSource" ref="dataSource"/>
        </bean>

</beans>
```

- Here we use *Jakarta Commons Database Connection Pools (DBCP)* to configure the datasource. The *BasicDataSource* can be easily configured and supports connection pooling. To use *DBCP* you need to have the following jar file in the classpath *commons-dbcp.jar* and *commons-pool.jar*. After creating the datasource inject the datasource to the DAO class. In the DAO class we use this datasource to create the *JDBCTemplate* object.
- The following jar files are required to run the example. All the *JDBCTemplate* related files are located in the *org.springframework.jdbc-3.0.0.M3.jar* file and the all the *DataAccessException* related classes are located in the *org.springframework.transaction-3.0.0.M3.jar* file.

```
antlr-runtime-3.0
commons-logging-1.0.4
org.springframework.asm-3.0.0.M3
org.springframework.beans-3.0.0.M3
org.springframework.context-3.0.0.M3
org.springframework.context.support-3.0.0.M3
org.springframework.core-3.0.0.M3
org.springframework.expression-3.0.0.M3

org.springframework.jdbc-3.0.0.M3.jar
org.springframework.transaction-3.0.0.M3.jar

hsqldb.jar

commons-dbcp.jar
commons-pool.jar
```

- To execute the example run the following Main class.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Main {

	public static void main(String[] args) {
		ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");
		ForumDAO forumDAO = (ForumDAO) context.getBean("forumDAO");
		Forum springForum = new Forum(1,"Spring Forum", "Discuss
everything related to Spring");
		forumDAO.insertForum(springForum);
		System.out.println(forumDAO.selectForum(1));

	}

}
```