**Spring Data access model also called Spring JDBC**

Spring jdbc is an abstraction layer on top of jdbc technologies.With Jdbc technologies, We can transfer the text format of data between a java application to database.

While working with JDBC ,We had major problems.

1.Biolerplate code:-

It is nothing but repeated code .In jdbc application each time we have to write a code for the loading the driver ,opening a connection ,statement Resultset and then we have to close the resources at last .

So this is repeated code in application.

2.)Checked exception:-

The Exception of JDBC are checked exception .so in a project we have to write the code related to database connection and all inside the try catch block exception.

This increase the burden on java developer.

3.Memory leak problem:-

In JDBC, Opening a and closing the object should be taken care by the developer . if objects are not handled properly then memory leak problems occur. In order to overcome the above problem we got the Spring framework jdbc technologies which is nothing but abstraction layer on the existing JDBC technology.

Springframework has given the central class called JDBCTemplate which have some convenient methods to execute the query on database.

This class comes under the package of org.springframework.jdbc.code.JdbcTemplate.

In springframework, The opening and closing an object of JDBC taken care by the spring only.

In springframework all exception are unchecked exception so developer should not concentrate on writing try catch blocks.

There is no memory leak because of spring handle to closing and opening an object according to requirement.

In JDBC layer, programmer only concentrate on constructing efficient query to run on database.

The given example of jdbc and spring ,where we are not using JdbcTemplate.

In this application we are using spring to get Datasource object only but we are writing a boiler code each time to perform the operation on the database.

//insert a code here

In JDBC, We have two ways to obtain connection to database.

1.Using DriverManager class

2.using Datasource interface.

Spring always use Datasource Interface to get connection from Database.

In Spring JDBC layer,”JdbcTemplate “ class open and close the connection internally so JdbcTemplate class depends on Datasource object.

Datasource is an interface and we can configure its any implementation class in configuration file of spring.

Q.> How to inject the datasource object to JdbcTemplate.

Ans-In JdbcTemplate class bothe constructor and setter method given to inject the Datasource object.

Way 1

<Bean id=”jt” class =”org.springframework.jdbc.core.JdbcTemplate>

<constructor-arg ref = “ds”/>

</bean>

<Bean id = “ds” class=org.springframework.jdbc.datasource.DriverManagerDataSource>

</bean>

Way 2

<Bean id=”jt” class =”org.springframework.jdbc.core.JdbcTemplate>

<property name=”datasource” ref=”ds”/>

</bean>

<Bean id = “ds” class=org.springframework.jdbc.datasource.DriverManagerDataSource>

</bean>

**Method of JdbcTemplate class**

1.execute()

2.update()

3.queryForMap()

4.queryForList()

5.queryForObject()

6.queryForRowSet()

1.>execute()

This method is used to execute the DDL command only .A DDL command is always static command so we have a single syntax for execute the DDL command.

Execute(string sql)

e.g

jT.execute(“create table Student(sid number(5),name varchar2(12)));

2.>update()

This method is called to execute the static or dynamic DML command. Insert ,delete or update .

This method is for static and dynamic so it has two syntax.

Update(String sql)

Update(String sql,Object …. args)----dynamic

e.g jt.update(“insert into student values(?,?,?)”,id,marks,name)

3.queryForMap()(To select only one row )

This method is used to read only one row from the Database with select operation.

Internally Springframework converts the one selected row into ResultSet to Map object.

In this Map object, column name are used as keys. We can run either static or dynmic query.

queryForMap(String sql)

queryForMap(String sql,Object----args)

e.g

Map map = jt.queryForMap(Select \* from emp where empno=1234”);

Set set = map.entrySet();

Iterator it = set.iterator();

While(it.hasNext()){

Map.Entry entry = (Map.entry)it.next()

Sop(entry.getKey()+”” +entry.getValue());

}

Or

For(Map.Entry me :map.entrySet()){

Sop(me.getKey()+”” + me.getValue());

}

Extract only key and values in different way

Map<Integer,Integer> map = new HashMap<Integer,Integer>();

//iterating over key only

For(Integer key :map.keySet())

{

Sysout(Key);

}

For(Integer value:map.values()){

Sysout(value);

}

Or Using generic way

Map<Integer,Integer> map = new HashMap<Integer,Integer>();

Iterator<Map.Entry<Integer,Integer>> me = map.entrySet().iterator();

While(me.hasNext()){

Map.Entry<Integer,Integer> entry = me.next();

Sysout(entry.getKey+””+””+entry.getvalue());

}

4.>queryForList()(to select multiple row or specific row)

This method is used when we want to select multiple rows from database.Internally Springframework converts ResultSet object for each row into Map object and stores it into List object.

Collection type only List allowed .

List list = jt.queryForList(Select \* from emp where deptno=20”);

Here List object contains the Map object so we can apply logic here to extract all map object like above.

Iterator it = list.iterator();

While(it.hasNext()){

Map map = (Map)it.next()

}

queryForList() supports both static and dynamic queries also.

5.)QueryForObject():-

This method works as queryForMap() to select one row from database

here we are specifying a required class type to convert row of ResultSet object to queryForObject(). But in case of queryForMap() it will return map object . But queryForObject() will return class type object.

Format

Employee emp = jt.queryForObject(String sql,Class clazz)/jt.queryForObject(String sql,class clazz,Object------args);

e.g.

Employee emp =jt.queryForObject(“Select \* from emp where empno=12”,Employee.class)

This method can be use for both static and dynamic.

6.)queryForRowSet();