RowMapper Example | Fetching records by Spring JdbcTemplate

Like ResultSetExtractor, we can use RowMapper interface to fetch the records from the database using **query()** method of **JdbcTemplate** class. In the execute of we need to pass the instance of RowMapper now.

Syntax of query method using RowMapper

```
public T query(String sql,RowMapper<T> rm)
```

RowMapper Interface

RowMapper interface allows to map a row of the relations with the instance of user-defined class. It iterates the ResultSet internally and adds it into the collection. So we don't need to write a lot of code to fetch the records as ResultSetExtractor.

Advantage of RowMapper over ResultSetExtractor

RowMapper saves a lot of code becuase it internally adds the data of ResultSet into the collection.

Method of RowMapper interface

It defines only one method mapRow that accepts ResultSet instance and int as the parameter list. Syntax of the method is given below:

```
public T mapRow(ResultSet rs, int rowNumber)throws SQLException
```

Example of RowMapper Interface to show all the records of the table

We are assuming that you have created the following table inside the Oracle10g database.

```
create table employee(
id number(10),
name varchar2(100),
salary number(10)
);
```

Employee.java

This class contains 3 properties with constructors and setter and getters and one extra method toString().

```
package com.javatpoint;

public class Employee {
private int id;
private String name;
private float salary;
//no-arg and parameterized constructors
//getters and setters
public String toString(){
    return id+" "+name+" "+salary;
}
}
```

EmployeeDao.java

It contains on property jdbcTemplate and one method getAllEmployeesRowMapper.

```
package com.javatpoint;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.core.RowMapper;

public class EmployeeDao {
private JdbcTemplate template;
public void setTemplate(JdbcTemplate template) {
    this.template = template;
}

public List<Employee> getAllEmployeesRowMapper(){
 return template.query("select * from employee",new RowMapper<Employee>(){
    @Override
    public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
        Employee e=new Employee();
        e.setId(rs.getInt(1));
        e.setName(rs.getString(2));
        e.setSalary(rs.getInt(3));
        return e;
    }
    });
}
}
```

applicationContext.xml

The **DriverManagerDataSource** is used to contain the information about the database such as driver class name, connnection URL, username and password.

There are a property named **datasource** in the JdbcTemplate class of DriverManagerDataSource type. So, we need to provide the reference of DriverManagerDataSource object in the JdbcTemplate class for the datasource property.

Here, we are using the JdbcTemplate object in the EmployeeDao class, so we are passing it by the setter method but you can use constructor also.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
```

```
<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
<property name="username" value="system" />
<property name="password" value="oracle" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="edao" class="com.javatpoint.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

</beans>
```

Test.java

This class gets the bean from the applicationContext.xml file and calls the getAllEmployeesRowMapper() method of EmployeeDao class.

```
package com.javatpoint;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {
public static void main(String[] args) {
    ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
    EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
    List<Employee> list=dao.getAllEmployeesRowMapper();

    for(Employee e:list)
        System.out.println(e);
}
}
```

download this example (developed using MyEclipse IDE)
download this example (developed using Eclipse IDE)

<<prev next>>

Please Share

Learn Latest Tutorials

Compiler D. JDB

Polymer SVG

⇧