MYBATIS - UPDATE OPERATION

We discussed, in the last chapter, how to perform READ operation on a table using MyBatis. This chapter explains how you can update records in a table using it.

We have the following STUDENT table in MySQL −

```
CREATE TABLE details.student(
   ID int(10) NOT NULL AUTO_INCREMENT,
   NAME varchar(100) NOT NULL,
   BRANCH varchar(255) NOT NULL,
   PERCENTAGE int(3) NOT NULL,
   PHONE int(11) NOT NULL,
   EMAIL varchar(255) NOT NULL,
   PRIMARY KEY (`ID`)
);
```

Assume this table has two record as follows −

```
mysql> select * from STUDENT;
+----+----------+--------+------------+-----------+--------------------+
| ID |   NAME   | BRANCH | PERCENTAGE |   PHONE   |      EMAIL         |
+----+----------+--------+------------+-----------+--------------------+
| 1  | Mohammad |   It   |     80     | 984803322 | Mohammad@gmail.com |
| 2  | shyam    |   It   |     75     | 984800000 | shyam@gmail.com    |
+----+----------+--------+------------+-----------+--------------------+
```

Student POJO Class

To perform update operation, you would need to modify Student.java file as −

```
public class Student {

   private int id;
   private String name;
   private String branch;
   private int percentage;
   private int phone;
   private String email;

   public Student(int id, String name,  String branch, int percentage, int phone, String
email) {
      super();
      this.id = id;
      this.name = name;
      this.setBranch(branch);
      this.setPercentage(percentage);
      this.phone = phone;
      this.email = email;
   }

   public Student() {}

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName() {
      return name;
```

```
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getPhone() {
      return phone;
   }

   public void setPhone(int phone) {
      this.phone = phone;
   }

   public String getEmail() {
      return email;
   }

   public void setEmail(String email) {
      this.email = email;
   }

   public String getBranch() {
      return branch;
   }

   public void setBranch(String branch) {
      this.branch = branch;
   }

   public int getPercentage() {
      return percentage;
   }

   public void setPercentage(int percentage) {
      this.percentage = percentage;
   }

   public String toString(){
      StringBuilder sb = new StringBuilder();

      sb.append("Id = ").append(id).append(" - ");
      sb.append("Name = ").append(name).append(" - ");
      sb.append("Branch = ").append(branch).append(" - ");
      sb.append("Percentage = ").append(percentage).append(" - ");
      sb.append("Phone = ").append(phone).append(" - ");
      sb.append("Email = ").append(email);

      return sb.toString();
   }

}
```

Student.xml File

To define SQL mapping statement using MyBatis, we would add **<update>** tag in Student.xml and inside this tag definition, we would define an **"id"** which will be used in mybatisUpdate.java file for executing SQL UPDATE query on database.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace = "Student">
   <resultMap id = "result" type = "Student">
      <result property = "id" column = "ID"/>
      <result property = "name" column = "NAME"/>
```

```
      <result property = "branch" column = "BRANCH"/>
      <result property = "percentage" column = "PERCENTAGE"/>
      <result property = "phone" column = "PHONE"/>
      <result property = "email" column = "EMAIL"/>
   </resultMap>

   <select id = "getById" parameterType = "int" resultMap = "result">
      SELECT * FROM STUDENT WHERE ID = #{id};
   </select>

   <update id = "update" parameterType = "Student">
      UPDATE STUDENT SET NAME = #{name},
         BRANCH = #{branch},
         PERCENTAGE = #{percentage},
         PHONE = #{phone},
         EMAIL = #{email}
      WHERE ID = #{id};
   </update>

</mapper>
```

mybatisUpdate.java File

This file has application level logic to update records into the Student table −

```
import java.io.IOException;
import java.io.Reader;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class mybatisUpdate {

   public static void main(String args[]) throws IOException{

      Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
      SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);
      SqlSession session = sqlSessionFactory.openSession();

      //select a particular student using id
      Student student = (Student) session.selectOne("Student.getById", 1);
      System.out.println("Current details of the student are" );
      System.out.println(student.toString());

      //Set new values to the mail and phone number of the student
      student.setEmail("mohamad123@yahoo.com");
      student.setPhone(90000000);

      //Update the student record
      session.update("Student.update",student);
      System.out.println("Record updated successfully");
      session.commit();
      session.close();

      //verifying the record
      Student std = (Student) session.selectOne("Student.getById", 1);
      System.out.println("Details of the student after update operation" );
      System.out.println(std.toString());
      session.commit();
      session.close();

   }
}
```

Compilation and Run

Here are the steps to compile and run mybatisUpdate.java. Make sure, you have set PATH and CLASSPATH appropriately before proceeding for compilation and execution.

- Create Student.xml as shown above.
- Create SqlMapConfig.xml as shown in the MYBATIS - Configuration XML chapter of this tutorial.
- Create Student.java as shown above and compile it.
- Create mybatisUpdate.java as shown above and compile it.
- Execute mybatisUpdate binary to run the program.

You would get following result. You can see the details of a particular record initially, and that record would be updated in STUDENT table and later, you can also see the updated record.

```
Current details of the student are
Id = 1 - Name = Mohammad - Branch = It - Percentage = 80 - Phone = 984802233 - Email =
mohammad@gmail.com
Record updated successfully
Details of the student after update operation
Id = 1 - Name = Mohammad - Branch = It - Percentage = 80 - Phone = 90000000 - Email =
mohamad123@yahoo.com
```

If you check the STUDENT table, it should display the following result −

```
mysql> select * from student;
+----+----------+--------+------------+-----------+----------------------+
| ID |  NAME    | BRANCH | PERCENTAGE |  PHONE    |       EMAIL          |
+----+----------+--------+------------+-----------+----------------------+
| 1  | Mohammad |  It    |     80     |  90000000 | mohamad123@yahoo.com |
| 2  | shyam    |  It    |     75     | 984800000 | shyam@gmail.com      |
+----+----------+--------+------------+-----------+----------------------+
2 rows in set (0.00 sec)
```