Patrón DAO

El patrón de objeto de acceso a datos o patrón DAO se utiliza para separar las operaciones o la API de acceso a datos de bajo nivel de los servicios comerciales de alto nivel. Los siguientes son los participantes en el patrón de objetos de acceso a datos.

* **Interfaz de objeto de acceso a datos** : esta interfaz define las operaciones estándar que se realizarán en un objeto modelo.
* **Clase concreta de objeto de acceso a datos** : esta clase implementa la interfaz anterior. Esta clase es responsable de obtener datos de una fuente de datos que puede ser una base de datos/xml o cualquier otro mecanismo de almacenamiento.
* **Objeto modelo u objeto de valor** : este objeto es un POJO simple que contiene métodos get/set para almacenar datos recuperados mediante la clase DAO.

Implementación

Vamos a crear un objeto *de Estudiante* que actúe como un Modelo o un Objeto de Valor. *StudentDao* es la interfaz de objetos de acceso a datos. *StudentDaoImpl* es una clase concreta que implementa la interfaz de objetos de acceso a datos. *DaoPatternDemo* , nuestra clase de demostración, utilizará *StudentDao* para demostrar el uso del patrón de objetos de acceso a datos.



Paso 1

Crear objeto de valor.

*Estudiante.java*

public class Student {

private String name;

private int rollNo;

Student(String name, int rollNo){

this.name = name;

this.rollNo = rollNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getRollNo() {

return rollNo;

}

public void setRollNo(int rollNo) {

this.rollNo = rollNo;

}

}

Paso 2

Crear interfaz de objetos de acceso a datos.

*StudentDao.java*

import java.util.List;

public interface StudentDao {

public List<Student> getAllStudents();

public Student getStudent(int rollNo);

public void updateStudent(Student student);

public void deleteStudent(Student student);

}

Paso 3

Cree una clase concreta que implemente la interfaz anterior.

*StudentDaoImpl.java*

import java.util.ArrayList;

import java.util.List;

public class StudentDaoImpl implements StudentDao {

//list is working as a database

List<Student> students;

public StudentDaoImpl(){

students = new ArrayList<Student>();

Student student1 = new Student("Robert",0);

Student student2 = new Student("John",1);

students.add(student1);

students.add(student2);

}

@Override

public void deleteStudent(Student student) {

students.remove(student.getRollNo());

System.out.println("Student: Roll No " + student.getRollNo() + ", deleted from database");

}

//retrive list of students from the database

@Override

public List<Student> getAllStudents() {

return students;

}

@Override

public Student getStudent(int rollNo) {

return students.get(rollNo);

}

@Override

public void updateStudent(Student student) {

students.get(student.getRollNo()).setName(student.getName());

System.out.println("Student: Roll No " + student.getRollNo() + ", updated in the database");

}

}

Etapa 4

Use *StudentDao* para demostrar el uso del patrón de objetos de acceso a datos.

*DaoPatternDemo.java*

public class DaoPatternDemo {

public static void main(String[] args) {

StudentDao studentDao = new StudentDaoImpl();

//print all students

for (Student student : studentDao.getAllStudents()) {

System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName() + " ]");

}

//update student

Student student =studentDao.getAllStudents().get(0);

student.setName("Michael");

studentDao.updateStudent(student);

//get the student

studentDao.getStudent(0);

System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " + student.getName() + " ]");

}

}

Paso 5

Verifique la salida.

Student: [RollNo : 0, Name : Robert ]

Student: [RollNo : 1, Name : John ]

Student: Roll No 0, updated in the database

Student: [RollNo : 0, Name : Michael ]