Spring + Hibernate IntegrationCreate table Student with following columns(rollno,name,sub1,sub2,sub3,sub4)Perform basic CRUD operations also Calculate total of all Subjects and PercentageAnd display Students Details In tabular format.

package com.app.spring.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import com.app.spring.model.StudentMode;

import com.app.spring.service.StudentService;

@Controller

@RequestMapping("/Student")

public class StudentController {

// need to inject our customer service

@Autowired

private StudentService customerService;

@GetMapping("/list")

public String listCustomers(Model theModel) {

// get customers from the service

List<StudentMode> theCustomers = customerService.getStudent();

// add the customers to the model

theModel.addAttribute("customers", theCustomers);

return "list-customers";

}

@GetMapping("/showFormForAdd")

public String showFormForAdd(Model theModel) {

// create model attribute to bind form data

StudentMode theCustomer = new StudentMode();

theModel.addAttribute("customer", theCustomer);

return "customer-form";

}

@PostMapping("/saveCustomer")

public String saveCustomer(@ModelAttribute("customer") StudentMode theCustomer) {

// save the customer using our service

customerService.saveStudent(theCustomer);

return "redirect:/customer/list";

}

@GetMapping("/showFormForUpdate")

public String showFormForUpdate(@RequestParam("customerId") int theId, Model theModel) {

// get the customer from our service

StudentMode theCustomer = customerService.getStudent(theId);

// set customer as a model attribute to pre-populate the form

theModel.addAttribute("customer", theCustomer);

// send over to our form

return "customer-form";

}

@GetMapping("/remove")

public String removeCustomer(@RequestParam("customerId") int theId) {

customerService.removeStudent(theId);

// send over to our form

return "redirect:/customer/list";

}

}

package com.app.spring.dao;

import java.util.List;

import com.app.spring.model.StudentMode;

public interface StudentDAO {

public List<StudentMode> getStudent();

public void saveStudent(StudentMode student);

public StudentMode getStudent(int rn);

public void removeStudent(int rn);

}

Step 3 .

package com.app.spring.dao;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.Query;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.app.spring.model.StudentMode;

@Repository

public class StudentDAOImpl implements StudentDAO {

// need to inject the session factory

@Autowired

private SessionFactory sessionFactory;

@Override

public List<StudentMode> getStudent(){

Session currentSession = sessionFactory.getCurrentSession();

Query<StudentMode> theQuery = currentSession.createQuery("from Student order by rn", StudentMode.class);

// execute query and get result list

List<StudentMode> student = theQuery.getResultList();

// return the results

return student;

}

@Override

public void saveStudent(StudentMode student) {

// get current hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// save/upate the customer ... finally LOL

currentSession.saveOrUpdate(student);

}

@Override

public StudentMode getStudent(int rn) {

// get the current hibernate session

Session currentSession = sessionFactory.getCurrentSession();

// now retrieve/read from database using the primary key

StudentMode student1 = currentSession.get(StudentMode.class, rn);

return student1;

}

@Override

public void removeStudent(int rn)

Session currentSession = sessionFactory.getCurrentSession();

// currentSession.delete(stud);

}

}

package com.app.spring.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="STUDENT")

public class StudentMode implements Serializable

{

/\*\*

\* serialVersionUID

\*/

private static final long serialVersionUID = 8633415090390966715L;

@Id

@Column(name="ID")

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

@Column(name="STUDENT\_NAME")

private String studentName;

@Column(name="ROLL\_NUMBER")

private int rollNumber;

@Column(name="COURSE")

private String course;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

package com.app.spring.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.app.spring.dao.StudentDAO;

import com.app.spring.model.StudentMode;

@Service

public class StudentService implements StudentService1 {

@Autowired

private StudentDAO customerDAO;

@Override

@Transactional

public List<StudentMode> getStudent() {

return customerDAO.getStudent();

}

@Override

@Transactional

public void saveStudent(StudentMode student) {

customerDAO.saveStudent(student);

}

@Override

@Transactional

public StudentMode getStudent(int rn) {

return customerDAO.getStudent(rn);

}

@Override

@Transactional

public void removeStudent(int rn) {

customerDAO.removeStudent(rn);

}

}

package com.app.spring.service;

import java.util.List;

import com.app.spring.model.StudentMode;

public interface StudentService1 {

public List<StudentMode> getStudent();

public void saveStudent(StudentMode student);

public StudentMode getStudent(int rn);

public void removeStudent(int rn);

}

Hibernate

Step 1.

package Student;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Table(name="STUDENT")

public class Student implements Serializable

{

private static final long serialVersionUID = 8633415090390966715L;

@Id

@Column(name="ID")

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

@Column(name="STUDENT\_NAME")

private String studentName;

@Column(name="ROLL\_NUMBER")

private int rollNumber;

@Column(name="COURSE")

private String Course;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

public String getcourse () {

return course;

}

public void setCourse(String course) {

this. course = course;

}

}

Step 2: Create Hibernate Configuration file

hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/hibernateDB2</property>

<property name="connection.username">root</property>

<property name="connection.password">pravin@1206</property>

<!-- JDBC connection pool (use the built-in) -->

<property name="connection.pool\_size">1</property>

<!-- SQL dialect -->

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Enable Hibernate's automatic session context management -->

<property name="current\_session\_context\_class">thread</property>

<!-- Disable the second-level cache -->

<property name="cache.provider\_class">org.hibernate.cache.NoCacheProvider</property>

<!-- Echo all executed SQL to stdout -->

<property name="show\_sql">true</property>

<!-- Drop and re-create the database schema on startup -->

<property name="hbm2ddl.auto">update</property>

<mapping class="com.sdnext.hibernate.tutorial.dto.Student">

</mapping></session-factory>

</hibernate-configuration>

Student.java

package com.sdnext.hibernate.tutorial.dto;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Table(name="STUDENT")

public class Student implements Serializable

{

/\*\*

\* serialVersionUID

\*/

private static final long serialVersionUID = 8633415090390966715L;

@Id

@Column(name="ID")

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

@Column(name="STUDENT\_NAME")

private String studentName;

@Column(name="ROLL\_NUMBER")

private int rollNumber;

@Column(name="COURSE")

private String course;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

Step 2: Create Hibernate Configuration file

hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/hibernateDB2</property>

<property name="connection.username">root</property>

<property name="connection.password">root</property>

<!-- JDBC connection pool (use the built-in) -->

<property name="connection.pool\_size">1</property>

<!-- SQL dialect -->

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Enable Hibernate's automatic session context management -->

<property name="current\_session\_context\_class">thread</property>

<!-- Disable the second-level cache -->

<property name="cache.provider\_class">org.hibernate.cache.NoCacheProvider</property>

<!-- Echo all executed SQL to stdout -->

<property name="show\_sql">true</property>

<!-- Drop and re-create the database schema on startup -->

<property name="hbm2ddl.auto">update</property>

<mapping class="com.sdnext.hibernate.tutorial.dto.Student">

</mapping></session-factory>

</hibernate-configuration>

Step 3: Create Hibernate Utility Class

HibernateUtil.java

package Student;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil

{

private static final SessionFactory sessionFactory;

static

{

try

{

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

}

catch(Throwable th){

System.err.println("Enitial SessionFactory creation failed"+th);

throw new ExceptionInInitializerError(th);

}

}

public static SessionFactory getSessionFactory(){

return sessionFactory;

}

}

CRUD Operations Using Hibernate (Annotation and Configuration)

CRUD Operations Using Hibernate (Annotation and Configuration)

In this section, you will learn how to develop a CRUD application using hibernate annotation.

Follows the following steps for developing the CRUD application in hibernate annotation.

Step 1: Create Domain Entity Class

Advertisements

Student.java

package com.sdnext.hibernate.tutorial.dto;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Table(name="STUDENT")

public class Student implements Serializable

{

/\*\*

\* serialVersionUID

\*/

private static final long serialVersionUID = 8633415090390966715L;

@Id

@Column(name="ID")

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

@Column(name="STUDENT\_NAME")

private String studentName;

@Column(name="ROLL\_NUMBER")

private int rollNumber;

@Column(name="COURSE")

private String course;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

Step 2: Create Hibernate Configuration file

hibernate.cfg.xml

file contains

(a.) database connection setting (database driver (com.mysql.jdbc.Driver), url (jdbc:mysql://localhost:3306/hibernateDB2), username (root) and password (root)),

(b.) SQL dialect (dialect – org.hibernate.dialect.MySQLDialect),

(c.) enable hibernate’s automatic session context management (current\_session\_context\_class – thread),

(d.) disable the second level cache (cache.provider\_class – org.hibernate.cache.NoCacheProvider),

(e.) print all executed SQL to stdout (show\_sql – true) and

(f.) drop and re-create the database schema on startup (hbm2ddl.auto – none).

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/hibernateDB</property>

<property name="connection.username">root</property>

<property name="connection.password">pravin@1206</property>

<!-- JDBC connection pool (use the built-in) -->

<property name="connection.pool\_size">1</property>

<!-- SQL dialect -->

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- Enable Hibernate's automatic session context management -->

<property name="current\_session\_context\_class">thread</property>

<!-- Disable the second-level cache -->

<property name="cache.provider\_class">org.hibernate.cache.NoCacheProvider</property>

<!-- Echo all executed SQL to stdout -->

<property name="show\_sql">true</property>

<!-- Drop and re-create the database schema on startup -->

<property name="hbm2ddl.auto">update</property>

<mapping class="com.sdnext.hibernate.tutorial.dto.Student">

</mapping></session-factory>

</hibernate-configuration>

Step 3: Create Hibernate Utility Class

HibernateUtil.java

package com.sdnext.hibernate.tutorial.utility;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil

{

private static final SessionFactory sessionFactory;

static

{

try

{

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

}

catch(Throwable th){

System.err.println("Enitial SessionFactory creation failed"+th);

throw new ExceptionInInitializerError(th);

}

}

public static SessionFactory getSessionFactory(){

return sessionFactory;

}

}

Step 4: Create Student on the database.

CreateStudent.java

package Student;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.sdnext.hibernate.tutorial.dto.Student;

import com.sdnext.hibernate.tutorial.utility.HibernateUtil;

public class CreateStudent {

public static void main(String[] args)

{

//Create student entity object

Student student = new Student();

student.setStudentName("Dinesh Rajput");

student.setRollNumber(01);

student.setSubject("MCA");

Student student = new Student();

student.setStudentName("Anamika Rajput");

student.setRollNumber(02);

student.setSubject("PGDAC");

Student student = new Student();

student.setStudentName("Adesh Rajput");

student.setRollNumber(03);

student.setSubject("MA");

Student student = new Student();

student.setStudentName("Vinesh Rajput");

student.setRollNumber(04);

student.setSubject("BA");

//Create session factory object

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

Session session = sessionFactory.openSession();

session.beginTransaction();

session.save(student);

System.out.println("Inserted Successfully");

session.getTransaction().commit();

session.close();

sessionFactory.close();

}

}

.close();

}

}