*** Permanent Data Store CRUD ***

*** Using Get, Post, Put, Delete and Patch ***

TheKiranAcademy
GopalSing

```
package com.Gopal;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
public class StudentController {

	@Autowired
	SessionFactory sf;

	@RequestMapping("API")
	public String myName() {
		return "Gopal";
	}

	@GetMapping("showSingleRecord/{id}")
	public Student one(@PathVariable int id) {
		Session s = sf.openSession();
		Student ss = s.get(Student.class, id);
		return ss;
	}

	@GetMapping("showMultipleRecords")
	public List<Student> all() {
		Session s = sf.openSession();
		Query query = s.createQuery("from Student");
		List<Student> list = query.list();
		return list;
	}
```

```
@PostMapping("insertSingleRecord")
public Student insert(@RequestBody Student student) {
	Session s = sf.openSession();
	Transaction t = s.beginTransaction();
	s.save(student);
	t.commit();
	return student;
}


@PostMapping("insertSingleRecordPath/{id}/{name}")
public Student insertAPI(@PathVariable int id, @PathVariable String name)
{
	Session s = sf.openSession();
	Transaction t = s.beginTransaction();
	Student ss = new Student(id, name);
	s.save(ss);
	t.commit();
	return ss;
}

@PostMapping("insertMultipleRecords")
public List<Student> multipleinsert(@RequestBody List<Student> list) {
	for (Student student : list) {
		Session ss = sf.openSession();
		Transaction tt = ss.beginTransaction();
		ss.save(student);
		tt.commit();
	}
	return list;
}
```

```
@PutMapping("updateSingleRecord")
public Student updateStudent(@RequestBody Student student) {
	Session ss = sf.openSession();
	Transaction tx = ss.beginTransaction();
	ss.update(student);
	System.out.println("Record Updated Successfully Done... " + student);
	tx.commit();
	return student;
}


@PutMapping("updateMultipleRecord")
public List<Student> multipleupdate(@RequestBody List<Student> list) {
	for (Student student : list) {
		Session ss = sf.openSession();
		Transaction tt = ss.beginTransaction();
		ss.update(student);
		tt.commit();
	}
	return list;
}
```

```
    @PatchMapping("partialSingleFieldUpdate/{id}/{name}")
    public Student partialSingleFieldUpdate(@PathVariable int id,
@PathVariable String name) {
        Session s = sf.openSession();
        Student student = s.get(Student.class, id);
        student.setName(name);
        Transaction tx = s.beginTransaction();
        s.update(student);
        tx.commit();
        return student;
    }


    @PatchMapping("partialMultipleFieldUpdate/{id}/{name}")
    public Student partialMultipleFieldUpdate(@PathVariable int id,
@PathVariable String name, @PathVariable String city) {
        Session s = sf.openSession();
        Student student = s.get(Student.class, id);
        student.setName(name);
        //student.setCity(city);
        Transaction tx = s.beginTransaction();
        s.update(student);
        tx.commit();
        return student;
    }
```

```
    @DeleteMapping("deleteSingleRecord/{id}")
    public int deleteStudent(@PathVariable int id) {
        Session ss = sf.openSession();
        Transaction tx = ss.beginTransaction();
        Student stud = ss.load(Student.class, id);
        ss.delete(stud);
        System.out.println("Record Deleted Successfully Done... " + id);
        tx.commit();
        return id;
    }


    @DeleteMapping("deleteMultipleRecord")
    public String deleteAll() {
        Session s = sf.openSession();
        Transaction t = s.beginTransaction();
        Query query = s.createQuery("DELETE FROM Student");
        query.executeUpdate();
        t.commit();
        return "Deletes All Table Record's";
    }

}
```