**Java Practical Assessment-4**

**THEORY QUESTIONS:**

**Q1. What are the major features in different versions of Spring Framework?**

**Ans:**

**1.Version Spring 2.5**

**2.Spring 3.0**

**3.Spring 4.0**

Feature This version was released in 2007. It was the first version which supported annotations. This version was released in 2009. It made full-fledged use of improvements in Java5 and also provided support to JEE6. This version was released in 2013. This was the first version to provide full support to Java 8.

**Q2. What is a Spring Framework?**

**Ans:** Spring is a powerful open source, application framework created to reduce the complexity of enterprise application development. It is light-weighted and loosely coupled. It has layered architecture, which allows you to select the components to use, while also providing a cohesive framework for J2EE application development. Spring framework is also called the framework of frameworks as it provides support to various other frameworks such as Struts, Hibernate, EJB, JSF etc.

**Q3. List the advantages of Spring Framework**

**Ans:**

**1.Spring is non-invasive:** That means you no need to implements any interface or inherit any class from spring to your classes, so whenever you want to change from spring to any other technology then you no need to change the logics of your class.

**2.Spring is light weight:** Spring is vast framework so spring people divide the whole spring in to different modules, they are designed in such a way that no module is dependent to other module, excep Spring core module, so according to your requirement you can learn a particular module, you no need to learn whole total framework.

**3.End to end Development:** Spring supports all aspects of application development, Business aspects, persistence aspects, etc, so we can develop a complete application using spring.

**4.Spring supports All types of application development:** We can develop any type of applications using spring, eg: Core java, web Application, Distributed application, Enterprise application.

**5.Spring is versatile:** We can integrate any technologies with spring, so we can say spring is versatile.

**6.Spring supports dependency injection:** The dependency between classes are managed by spring.

**Q4. What are the different features of Spring Framework?**

**Ans:** Following are some of the major features of Spring Framework:

**1.Lightweight:** Spring is lightweight when it comes to size and transparency. Inversion of control (IOC): The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion of Control.

**2.Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.

**3.Container:** Spring Framework creates and manages the life cycle and configuration of the application objects. MVC Framework: Spring Framework’s MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.

**4.Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework.

5.Spring’s transaction support can be also used in container less environments.

6.**JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

**Q5. How many modules are there in Spring Framework and what are they?**

**Ans:** There are around 20 modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test.

**1.Spring Core Container** – This layer is basically the core of Spring Framework. It contains the following modules: Spring Core Spring Bean SPEL (Spring Expression Language) Spring Context

**2.Data Access/Integration** – This layer provides support to interact with the database.

It contains the following modules:

1.JDBC (Java Database Connectivity)

2.ORM (Object Relational Mapping)

3.OXM (Object XML Mappers)

4.JMS (Java Messaging Service) Transactions.

**3.Web** – This layer provides support to create web application. It contains the following modules

**4.Web Web** – MVC Web – Socket Web – Portlet.

Aspect Oriented Programming (AOP) – In this layer you can use Advices, Pointcuts etc., to decouple the code. Instrumentation – This layer provides support to class instrumentation and class loader implementations. Test -This layer provides support to testing with JUnit and TestNG.

Few Miscellaneous modules are given below:

1.Messaging – This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.

2.Aspects – This module provides support to integration with AspectJ.

**Q6. What is a Spring configuration file?**

**Ans:** A Spring configuration file is an XML file. This file mainly contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and moreclean. If it’s not planned and written correctly, it becomes very difficult to manage in big projects.

**Q7. What are the different components of a Spring application?**

**Ans:** A Spring application, generally consists of following components:

1.**Interface**: It defines the functions.

2.**Bean class**: It contains properties, its setter and getter methods, functions etc.

3.**Spring Aspect Oriented Programming (AOP):** Provides the functionality of cross-cutting concerns.

4.**Bean Configuration File**: Contains the information of classes and how to configure then.

5.**User program**: It uses the function.

**Q8. What are the various ways of using Spring Framework?**

**Ans:** Spring Framework can be used in various ways. They are listed as follows:

1.As a Full-fledged Spring web application.

2.Asathirdpartywebframework, usingSpringFrameworksmiddle-tier.

3.For remote usage.

4.As Enterprise Java Bean which can wrap existing POJOs(PlainOldJava Objects).

**Q9. What is Spring IOC Container?**

**Ans:** At the core of the Spring Framework, lies the Spring container, the container creates the object, wires them together, configures them and manages their complete life cycle. The Spring container makes use of Dependency Injection to manage the components that make up an application. The container receives instructions for which objects to instantiate, configure, and assemble by reading the configuration metadata provided. This metadata can be provided either by XML, Java annotations or Java code.

**Q10. What do you mean by Dependency Injection?**

**Ans:** In Dependency Injection, you do not have to create your objects but have to describe how they should be created. You don’t connect your components and services together in the code directly, but describe which services are needed by which components in the configuration file. The IoC container will wire them up together.

**PRACTICAL QUESTIONS:**

**Q1. Write a code for YML configuration with jpa configuration?**

**Ans:**

server:

port: 8086  // Use this to specify the port on which you want to run the application

//DataBase Configuration

spring:

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver //Use this property to specify the driver

url: jdbc:mysql://localhost:3309/demo?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false

username: root //Mention your database username here

password: Mysql#803454 //Mention your database password here.

//JPA Configurationjpa:

database-platform: org.hibernate.dialect.MySQL8Dialect

generate-ddl: true

hibernate:

ddl-auto: update

show-sql: true

**Q2. Impliment Spring Actuator to monitor one microservice?**

**Ans:**

**SampleRestController:**

**package** com.example.demo;

**import** org.springframework.beans.factory.annotation.Value;

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

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** SampleRestController {

@GetMapping("/greeting1")

**public** String greeting1() {

**return** "Wel Come to Microservices";

}

// @Value("${myname}") placeholder

@Value("Venu Puppala")

**private** String name;

@GetMapping("/greeting")

**public** String greeting() {

**return** name + "Wel Come to Microservices";

}

}

**ActuatorDemoApplication:**

**package** com.example.demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** ActuatorDemoApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(ActuatorDemoApplication.**class**, args);

}

}

**Application.properties:**

server.port=7070

myname=venu puppala

management.endpoints.web.exposure.include=\*

server.error.whitelabel.enabled=false

management.endpoint.health.group.custom.include=diskSpace,ping

management.endpoint.health.group.custom.show-components=always

# |when\_authorized

management.endpoint.health.group.custom.show-details=always

#when\_authorized

management.endpoint.health.group.custom.status.http-mapping.up=207

management.endpoint.shutdown.enabled=true

management.endpoints.web.exposure.exclude=loggers

endpoints.beans.id=springbeans

endpoints.beans.sensitive=false

endpoints.beans.enabled=true

endpoints.health.sensitive=false

info.app.name=Spring Sample Application

info.app.description=This is my first spring boot application

info.app.version=1.0.0

#port used to expose actuator

management.port=8081

#CIDR allowed to hit actuator

management.address=127.0.0.1

#Whether security should be enabled or disabled altogether

management.security.enabled=false

security.user.name=admin

security.user.password=secret

management.security.role=SUPERUSER

**Q3. Please create and deploy one microservice using Spring CLI?**

**Ans:**

**Step 1**: Check Spring boot version

spring version

Spring CLI v2.6.2

**Step 2**: hello.groovy

This is my Controller

@RestController

class WebApplication {

@RequestMapping("/")

String home() {

"Hello World!"

}

}

**Step3**: To compile and run the application, type the following command:

spring run hello.groovy.

To set JVM command line arguments, you can use the JAVA\_OPTS environment variable, as shown

JAVA\_OPTS=-Xmx1024m spring run hello.groovy

**Q4. Please change default server port from 8080 to 9090?**

**Ans:** Specify the server port number in application.properties file as below

            Server.port =

            or if you have the application.yml file then mention the below

server:

  port: 9090

**Q5. How to resolve whitelabel error page in spring boot application?**

**Ans:**

Check for the @requestmapping path correctly and use that specified path only to

Run the operation. This should be the main cause of the error.

If we want to remove the whitelabel errors then we can do the following:

**Displaying custom page:**

Create a *error.html* page and put it into the **src/main/resources/templates** directory

**Using Application.properties method:**

#Disable Whitelabel Error Page server.error.whitelabel.enabled=false

**Q6. All Relationships example such as OneToOne, OneToMany, ManyTOne, ManyToMany using spring boot.**

**Ans:**

**OneToOne:**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OneToOnespringproject1Application {

public static void main(String[] args) {

SpringApplication.*run*(OneToOnespringproject1Application.class, args);

}

}

**Employee.java**

**package** com.example.demo;

**import** java.io.Serializable;

**import** java.util.Objects;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToOne;

@Entity

**public** **class** Employee **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** serialVersionUID = 4270767860636327928L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

**private** Long empID;

**private** String name, address, phone;

@OneToOne(cascade = CascadeType.ALL)

**private** Laptop laptop;

**public** Long getEmpID() {

**return** empID;

}

**public** **void** setEmpID(Long empID) {

**this**.empID = empID;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** Laptop getLaptop() {

**return** laptop;

}

**public** **void** setLaptop(Laptop laptop) {

**this**.laptop = laptop;

}

}

**Laptop.java:**

**package** com.example.demo;

**import** java.io.Serializable;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

@Entity

**public** **class** Laptop **implements** Serializable{

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 7144115798199856451L;

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long lid;

**private** String model,brand,cpu;

**public** Long getLid() {

**return** lid;

}

**public** **void** setLid(Long lid) {

**this**.lid = lid;

}

**public** String getModel() {

**return** model;

}

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** String getBrand() {

**return** brand;

}

**public** **void** setBrand(String brand) {

**this**.brand = brand;

}

**public** String getCpu() {

**return** cpu;

}

**public** **void** setCpu(String cpu) {

**this**.cpu = cpu;

}

}

**EmployeeRepository:**

**package** com.example.demo.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** com.example.demo.Employee;

@Repository

**public** **interface** EmployeeRepository **extends** JpaRepository<Employee, Long>{

}

**LaptopRepository:**

**package** com.example.demo.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** com.example.demo.Laptop;

@Repository

**public** **interface** LaptopRepository **extends** JpaRepository<Laptop, Long>{

}

**EmployeeRestController:**

**package** com.example.demo.restcontroller;

**import** java.util.List;

**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.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;

**import** com.example.demo.Employee;

**import** com.example.demo.repository.\*;

@RestController

@RequestMapping("/Employee")

**public** **class** EmpleoyeeRestController {

@Autowired

**private** EmployeeRepository employeeRepository;

@PostMapping("/")

**public** Employee addEmployee(@RequestBody Employee employee) {

**return** employeeRepository.save(employee);

}

@GetMapping("/")

**public** List<Employee> listEmployee() {

**return** employeeRepository.findAll();

}

@PutMapping("/")

**public** Employee updateEmployee(@RequestBody Employee employee) {

**return** employeeRepository.save(employee);

}

@DeleteMapping("/")

**public** String deleteEmployee(@RequestBody Employee employee) {

employeeRepository.delete(employee);

**return** "deleteddata";

}

}

**LaptopRestController:**

**package** com.example.demo.restcontroller;

**import** java.util.List;

**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.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;

**import** com.example.demo.Laptop;

**import** com.example.demo.repository.LaptopRepository;

@RestController

@RequestMapping("/laptop")

**public** **class** LaptopRestController {

@Autowired

**private** LaptopRepository laptopRepository;

@PostMapping("/")

**public** Laptop addLaptop(@RequestBody Laptop laptop) {

**return** laptopRepository.save(laptop);

}

@GetMapping("/")

**public** List<Laptop> listLaptop() {

**return** laptopRepository.findAll();

}

@PutMapping("/")

**public** Laptop updateLaptop(@RequestBody Laptop laptop) {

**return** laptopRepository.save(laptop);

}

@DeleteMapping("/")

**public** String deleteLaptop(@RequestBody Laptop laptop) {

laptopRepository.delete(laptop);

**return** "deleteddata";

}

}

**OneToMany:**

**package** com.example.demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** OneToManyApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.run(OneToManyApplication.**class**, args);

}

}

**Employee.java:**

**package** com.example.demo.Entity;

**import** java.io.Serializable;

**import** java.util.List;

**import** java.util.Objects;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

@Entity

**public** **class** Employee **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 7144115798199856451L;

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long eid;

**private** String name, address, phone;

@OneToMany(cascade = CascadeType.***ALL***)

**private** List<Phone> phones;

**public** Long getEid() {

**return** eid;

}

**public** **void** setEid(Long eid) {

**this**.eid = eid;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** List<Phone> getPhones() {

**return** phones;

}

**public** **void** setPhones(List<Phone> phones) {

**this**.phones = phones;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(address, eid, name, phone, phones);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Employee other = (Employee) obj;

**return** Objects.*equals*(address, other.address) && Objects.*equals*(eid, other.eid)

&& Objects.*equals*(name, other.name) && Objects.*equals*(phone, other.phone)

&& Objects.*equals*(phones, other.phones);

}

@Override

**public** String toString() {

**return** "Employee [eid=" + eid + ", name=" + name + ", address=" + address + ", phone=" + phone + ", phones="

+ phones + ", getEid()=" + getEid() + ", getName()=" + getName() + ", getAddress()=" + getAddress()

+ ", getPhone()=" + getPhone() + ", getPhones()=" + getPhones() + ", hashCode()=" + hashCode()

+ ", getClass()=" + getClass() + ", toString()=" + **super**.toString() + "]";

}

}

**PHONE\_TYPE:**

**package** com.example.demo.Entity;

**public** **enum** PHONE\_TYPE {

***HOME***, ***OFFICE***, ***IMMERGENCY***, ***PERMANENT***

}

**Phone.java:**

**package** com.example.demo.Entity;

**import** java.util.Objects;

**import** javax.persistence.Entity;

**import** javax.persistence.EnumType;

**import** javax.persistence.Enumerated;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToMany;

@Entity

**public** **class** Phone {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long pid;

**private** String comment, phoneNumber;

@Enumerated(EnumType.***STRING***)

**private** PHONE\_TYPE phone\_TYPE;

@ManyToMany

@JoinColumn(name = "employee\_id", nullable = **false**)

**private** Employee employee;

**public** Long getPid() {

**return** pid;

}

**public** **void** setPid(Long pid) {

**this**.pid = pid;

}

**public** String getComment() {

**return** comment;

}

**public** **void** setComment(String comment) {

**this**.comment = comment;

}

**public** String getPhoneNumber() {

**return** phoneNumber;

}

**public** **void** setPhoneNumber(String phoneNumber) {

**this**.phoneNumber = phoneNumber;

}

**public** PHONE\_TYPE getPhone\_TYPE() {

**return** phone\_TYPE;

}

**public** **void** setPhone\_TYPE(PHONE\_TYPE phone\_TYPE) {

**this**.phone\_TYPE = phone\_TYPE;

}

**public** Employee getEmployee() {

**return** employee;

}

**public** **void** setEmployee(Employee employee) {

**this**.employee = employee;

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(comment, employee, phoneNumber, phone\_TYPE, pid);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Phone other = (Phone) obj;

**return** Objects.*equals*(comment, other.comment) && Objects.*equals*(employee, other.employee)

&& Objects.*equals*(phoneNumber, other.phoneNumber) && phone\_TYPE == other.phone\_TYPE

&& Objects.*equals*(pid, other.pid);

}

@Override

**public** String toString() {

**return** "Phone [pid=" + pid + ", comment=" + comment + ", phoneNumber=" + phoneNumber + ", phone\_TYPE="

+ phone\_TYPE + ", employee=" + employee + "]";

}

}

**EmployeeRestController:**

**package** com.example.demo.Controller;

**import** java.util.List;

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

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

**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;

**import** com.example.demo.Entity.Employee;

**import** com.example.demo.repository.EmployeeRepository;

@RestController

@RequestMapping("/employee")

**public** **class** EmployeeRestController {

@Autowired

**private** EmployeeRepository employeeRepository;

@PostMapping("/")

**public** Employee addEmployee(@RequestBody Employee employee) {

**return** employeeRepository.save(employee);

}

@GetMapping("/")

**public** List<Employee> listEmployee() {

**return** employeeRepository.findAll();

}

@PutMapping("/")

**public** Employee updateEmployee(@RequestBody Employee employee) {

**return** employeeRepository.save(employee);

}

}

**PhoneController:**

**package** com.example.demo.Controller;

**import** java.util.List;

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

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

**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;

**import** com.example.demo.Entity.Phone;

**import** com.example.demo.repository.PhoneRepository;

@RestController

@RequestMapping("/phone")

**public** **class** PhoneController{

@Autowired

**private** PhoneRepository phoneRepository;

@PostMapping("/")

**public** Phone addPhone(@RequestBody Phone phone) {

**return** phoneRepository.save(phone);

}

@GetMapping("/")

**public** List<Phone> listPhone() {

**return** phoneRepository.findAll();

}

@PutMapping("/")

**public** Phone updatePhone(@RequestBody Phone phone) {

**return** phoneRepository.save(phone);

}

}

**Phone Repository:**

**package** com.example.demo.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** com.example.demo.Entity.Phone;

@Repository

**public** **interface** PhoneRepository **extends** JpaRepository<Phone, Long>{

}

**Employee Repository:**

**package** com.example.demo.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.example.demo.Entity.Employee;

**public** **interface** EmployeeRepository **extends** JpaRepository<Employee, Long>{

}

**ManyToMany:**

**Employee Class:**

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

@Entity

public class Employee implements Serializable {

/\*\*

\*

\*/

private static final long serialVersionUID = 7108602029108606198L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long eid;

private String name, address;

@ManyToMany(cascade = CascadeType.ALL)

@JoinTable(name = "Employee\_Project", joinColumns = { @JoinColumn(name = "employee\_eid") }, inverseJoinColumns = {

@JoinColumn(name = "project\_pid") })

private Set<Project> projects = new HashSet<Project>();

public Long getEid() {

return eid;

}

public void setEid(Long eid) {

this.eid = eid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public Set<Project> getProjects() {

return projects;

}

public void setProjects(Set<Project> projects) {

this.projects = projects;

}

}

**Project Class:**

import java.io.Serializable;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

@Entity

public class Project implements Serializable {

/\*\*

\*

\*/

private static final long serialVersionUID = 6116602877170941471L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long pid;

private String title;

@ManyToMany(cascade = CascadeType.ALL, mappedBy = "projects")

private Set<Employee> employees;

public Long getPid() {

return pid;

}

public void setPid(Long pid) {

this.pid = pid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Set<Employee> getEmployees() {

return employees;

}

public void setEmployees(Set<Employee> employees) {

this.employees = employees;

}

}

**Employee Repository:**

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.springboot\_hybernet\_many\_to\_many.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long>{

}

**Project Repository:**

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.springboot\_hybernet\_many\_to\_many.model.Project;

@Repository

public interface ProjectRepository extends JpaRepository<Project, Long>{

}

**Employee RestController:**

import java.util.List;

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.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;

import com.springboot\_hybernet\_many\_to\_many.model.Employee;

import com.springboot\_hybernet\_many\_to\_many.repository.EmployeeRepository;

@RestController

@RequestMapping("/employee")

public class EmployeeRestController {

@Autowired

private EmployeeRepository empRepo;

@PostMapping("/")

public Employee addEmployee(@RequestBody Employee emp) {

return empRepo.save(emp);

}

@GetMapping("/")

public List<Employee> listEmployee() {

return empRepo.findAll();

}

@PutMapping("/")

public Employee updateEmployee(@RequestBody Employee emp) {

return empRepo.save(emp);

}

@DeleteMapping("/")

public String deleteEmployee(@RequestBody Employee emp) {

empRepo.delete(emp);

return "Deleted Employee Data";

}

}

**Project RestController:**

import java.util.List;

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.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;

import com.springboot\_hybernet\_many\_to\_many.model.Project;

import com.springboot\_hybernet\_many\_to\_many.repository.ProjectRepository;

@RestController

@RequestMapping("/project")

public class ProjectRestController {

@Autowired

private ProjectRepository prjctRepo;

@PostMapping("/")

public Project addProject(@RequestBody Project prjt) {

return prjctRepo.save(prjt);

}

@GetMapping("/")

public List<Project> listProjects() {

return prjctRepo.findAll();

}

@PutMapping("/")

public Project updateProject(@RequestBody Project prjt) {

return prjctRepo.save(prjt);

}

@DeleteMapping("/")

public String deleteProject(@RequestBody Project prjt) {

prjctRepo.delete(prjt);

return "Project Data Deleted";

}

}