1. **Create REST APIs expose CRUD operation on following enities?**

```
CREATE TABLE `product` (
  `id` int(11) NOT NULL
AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `price` float NOT NULL,
  PRIMARY KEY (`id`)
);
```

<u>Pom.xml file</u>

```
<parent>

    <groupId>org.springframework.boot<
/groupId>

    <artifactId>spring-boot-starter-
parent</artifactId>

    <version>2.2.2.RELEASE</version>

</parent>


<dependencies>

    <dependency>

        <groupId>org.springframework.b
oot</groupId>
```

```
        <artifactId>spring-boot-
starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.b
oot</groupId>
        <artifactId>spring-boot-
starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-
java</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>

<properties>
    <java.version>1.8</java.version>
</properties>
```

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframewo
rk.boot</groupId>
            <artifactId>spring-boot-
maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Application.properties

```
spring.jpa.hibernate.ddl-auto=none
spring.datasource.url=jdbc:mysql://loc
alhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=password
```

package net.codejava;

```
package net.codejava;
```

```
import javax.persistence.Entity;
import
javax.persistence.GeneratedValue;
import
javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {
    private Integer id;
    private String name;
    private float price;

    public Product() {
    }

    public Product(Integer id, String
name, float price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    public Integer getId() {
        return id;
    }

    // other getters and setters...
}
```

ProductRepository

```
package net.codejava;

import
org.springframework.data.jpa.repositor
y.JpaRepository;
```

```
public interface
ProductRepository extends
JpaRepository<Product, Integer> {


}
```

<u>ProductServises</u>

```
package net.codejava;


import java.util.List;


import javax.transaction.Transactional;


import
org.springframework.beans.factory.anno
tation.Autowired;
import
org.springframework.stereotype.Service
;


@Service
@Transactional
```

```
public class ProductService {

    @Autowired
    private ProductRepository repo;

    public List<Product> listAll() {
        return repo.findAll();
    }

    public void save(Product product) {
        repo.save(product);
    }

    public Product get(Integer id) {
        return repo.findById(id).get();
    }

    public void delete(Integer id) {
        repo.deleteById(id);
    }
}
```

Springboot

```
package net.codejava;


import
org.springframework.boot.SpringApplica
tion;
import
org.springframework.boot.autoconfigure
.SpringBootApplication;


@SpringBootApplication
public class Application {


    public static void main(String[]
args) {
        SpringApplication.run(Applicat
ion.class, args);
    }


}
```

2. **Make use of bean validation to ensure data consistency**

Customer Java

```
Import Java.validation.constraints.Size;
Public class Customer{
@Positive
private Integer id;
@Not Blank
@Size(min=1,max=45)
private String name;
@Email
private String email;
@NotEmpty
private List<String>address;
public Customer (Integer id,String name, String
email, List<String>address) {
this.id=id;
this.name=name;
this.email=email;
this.address=address;
}
public List<String>getAddress(){
return address;
}}
```

CustomerTest Java

```
import
com.eldermoraes.beanvalidation.Customer;
import java.util.Arrays;
import java.util.Set;
import javax.validation.ConstraintViolation;
import javax.validation.Validation;
import javax.validation.Validation;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Text;
public class CustomerTest{
private static Validator VALIDATOR;
public CustomerTest(){
}
@Before
public void setUp(){
VALIDATOR
=Validation.buildDefaultValidatiorFactory().getVali
dator();
}
@Test
Public void validCustomer(){
Customer c=new
Cutomer(1,”Customer”,Customer@test.com,Arrays
, asList(new String []{“address 1”,”address2”}));
Set<ConstraintViolation<Customer>>cons=VALI
DATOR.validate(c);
```

```
Assert. assertTrue(cons.isEmpty());
}
@Test
Public void negativeId(){
Customer c=new Customer (-
11,"Customer",customer @test.com",Arrays.
asList(new String []{"address 1","address2"}));
Set<ConstraintViolation<Customer>>cons=VALI
DATOR.validate(c);
Assert.assertTrue(cons.size()==1);
}
@Test
Public void blankName(){
Customer c=new Customer
(1,"Customer",customer @test.com",Arrays.
asList(new String []{"address 1","address2"}));
Set<ConstraintViolation<Customer>>cons=VALI
DATOR.validate(c);
```

```
Assert.assertTrue(cons.size()==1);
}
@Test
Public void longName(){
Customer c=new Customer (-11,"Customer",customer @test.com",Arrays.asList(new String []{"address 1","address2"}));
Set<ConstraintViolation<Customer>>cons=VALIDATOR.validate(c);
Assert.assertTrue(cons.size()==1);
}
```

3.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xmlns="http://java.sun.com/xml/ns/
javaee"
    xmlns:web="http://java.sun.com/xml
/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.su
n.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/
web-app_3_0.xsd"
    version="3.0">

    <display-name>RESTful CRUD
Example</display-name>

    <servlet>
        <servlet-
name>jerseyServlet</servlet-name>
        <servlet-
class>com.sun.jersey.spi.container.ser
vlet.ServletContainer</servlet-class>
        <init-param>
```

```
            <param-
name>com.sun.jersey.config.property.pa
ckages</param-name>
            <param-
value>org.o7planning.restfulcrud</para
m-value>
        </init-param>
        <load-on-startup>1</load-on-
startup>
    </servlet>
    <servlet-mapping>
        <servlet-
name>jerseyServlet</servlet-name>
        <url-pattern>/rest/*</url-
pattern>
    </servlet-mapping>



</web-app>
```

Employee.java

```
package org.o7planning.restfulcrud.dao;
```

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.o7planning.restfulcrud.model.Employee;

public class EmployeeDAO {

    private static final Map<String, Employee> empMap = new
HashMap<String, Employee>();

    static {
        initEmps();
    }

    private static void initEmps() {
        Employee emp1 = new Employee("E01", "Smith",
"Clerk");
        Employee emp2 = new Employee("E02", "Allen",
"Salesman");
        Employee emp3 = new Employee("E03", "Jones",
"Manager");

        empMap.put(emp1.getEmpNo(), emp1);
        empMap.put(emp2.getEmpNo(), emp2);
        empMap.put(emp3.getEmpNo(), emp3);
    }
    public static Employee getEmployee(String empNo) {
        return empMap.get(empNo);
    }

    public static Employee addEmployee(Employee emp) {
        empMap.put(emp.getEmpNo(), emp);
        return emp;
    }

    public static Employee updateEmployee(Employee emp) {
        empMap.put(emp.getEmpNo(), emp);
        return emp;
    }

    public static void deleteEmployee(String empNo) {
        empMap.remove(empNo);
    }

    public static List<Employee> getAllEmployees() {
        Collection<Employee> c = empMap.values();
        List<Employee> list = new ArrayList<Employee>();
        list.addAll(c);
        return list;
    }

    List<Employee> list;

}
```