09-Aug-2022 SPRING BOOT WITH MYSQL CONNECTION

****

PROCEDUIRE:

---

step 1:

#.first we need to create spring boot application named as MYSQLTESTSERVICE

step 2:

#.AFTER CREATING THE PROJECT THE EXECUTION STARTS FROM RESOURCE MAIN

The resource main consists of url,username,password and so.

resource main

****

```
spring.datasource.url=jdbc:mysql://localhost:3306/mukeshdb?useSSL=false
spring.datasource.username=root
spring.datasource.password=root
#spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

---

step 3:

#.At thired step the execution is starts from src/main:

which consists of main method with SpringApplication.run()

this method is the actual response of running our boot application

src/main

****

```
package com.mukesh;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MySqlTestSeviceApplication {

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

}
```

---

step 4:

#.com.mukesh.controller; in this work space we have to display our table and
queries based on our desire.

#.@Autowired-is used to call the repo interface

#.@GetMapping("greet")-is used to generate a url named as specified string in the method of GetMapping()
controller step

#.After that we need to create a method which return a string for our clarification when executing in the
browser
using this url http://localhost:8080/greet after hiting the url if it executes
without exception which display returned string in the browser

```
package com.mukesh.controller;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
```

```
import com.mukesh.model.Person;
import com.mukesh.repo.PersonRepo;

@org.springframework.web.bind.annotation.RestController
public class RestController {

    @Autowired
    public PersonRepo repo;

    @GetMapping("greet")
    public String greetUser() {

        Iterable<Person> persons = repo.findAll();    // for diplaying all the attributes in the table

        Optional<Person> user = repo.findById(-333); // optional is used to return either the value may contain or
        not but not null

        System.out.println("User of Id (-333) : " + user);


        System.out.println("All persons found: " + persons);

        return "Hi";
    }

    @GetMapping("user/{id}")     // is used to get the input from user at the browser if the entered input is
    present which
                                        display the respective attributes otherwise throw exception
    public Person greetUserById(@PathVariable("id")int id) {

        //Iterable<Person> persons = repo.findAll();
        System.out.println("Reseved value : " +id);
        Optional<Person> user = repo.findById(id);
```

```
        System.out.println("User of Id (-333) : " + user);


        //System.out.println("All persons found: " + persons);

        return user.orElseGet(null);
    }
}
```

---

step 5:

#.com.mukesh.repo; - it is an interface which extends the CrudRepository<Person, Integer>
                                                                |its our table
                                                                and intger is our primary key datatype

repo

```
package com.mukesh.repo;

import org.springframework.data.repository.CrudRepository;

import com.mukesh.model.Person;

public interface PersonRepo extends CrudRepository<Person, Integer> {

}
```

---

step 6: com.mukesh.model; - its our table name with attributes

---

model

****

```
package com.mukesh.model;

import javax.persistence.Entity;
```

```
import javax.persistence.Id;

@Entity   // defines the overall table usage
public class Person {
   @Id    // defines the type of primary key
   private int aadharno;
   private String firstname;
   private String lastname;
   private String city;
   private String job;

   @Override
   public String toString() {
      return "Person [aadharno=" + aadharno + ", firstname=" + firstname + ", lastname=" + lastname + ",
      city=" + city
            + ", job=" + job + "]";
   }
   public int getAadharno() {
      return aadharno;
   }
   public void setAadharno(int aadharno) {
      this.aadharno = aadharno;
   }
   public String getFirstname() {
      return firstname;
   }
   public void setFirstname(String firstname) {
      this.firstname = firstname;
   }
   public String getLastname() {
      return lastname;
   }
   public void setLastname(String lastname) {
      this.lastname = lastname;
   }
```

```
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getJob() {
        return job;
    }
    public void setJob(String job) {
        this.job = job;
    }

}
```

The springBoot JDBC displays the table in browser in the format of key value pair
ie JSON