Q1. Create a Restful API using Spring Boot for Student.

Solution

Student.java

```
package com.spring.springBootDemo.entities;

public class Student {
  private int rollNumber;
  private String name;
  private String course;

  public Student() {
  }

  public int getRollNumber() {
     return rollNumber;
  }

  public void setRollNumber(int rollNumber) {
     this.rollNumber = rollNumber;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public String getCourse() {
     return course;
  }

  public void setCourse(String course) {
     this.course = course;
  }

  public Student(int rollNumber, String name, String course) {
     this.rollNumber = rollNumber;
     this.name = name;
     this.course = course;
  }
}
```

StudentController.java

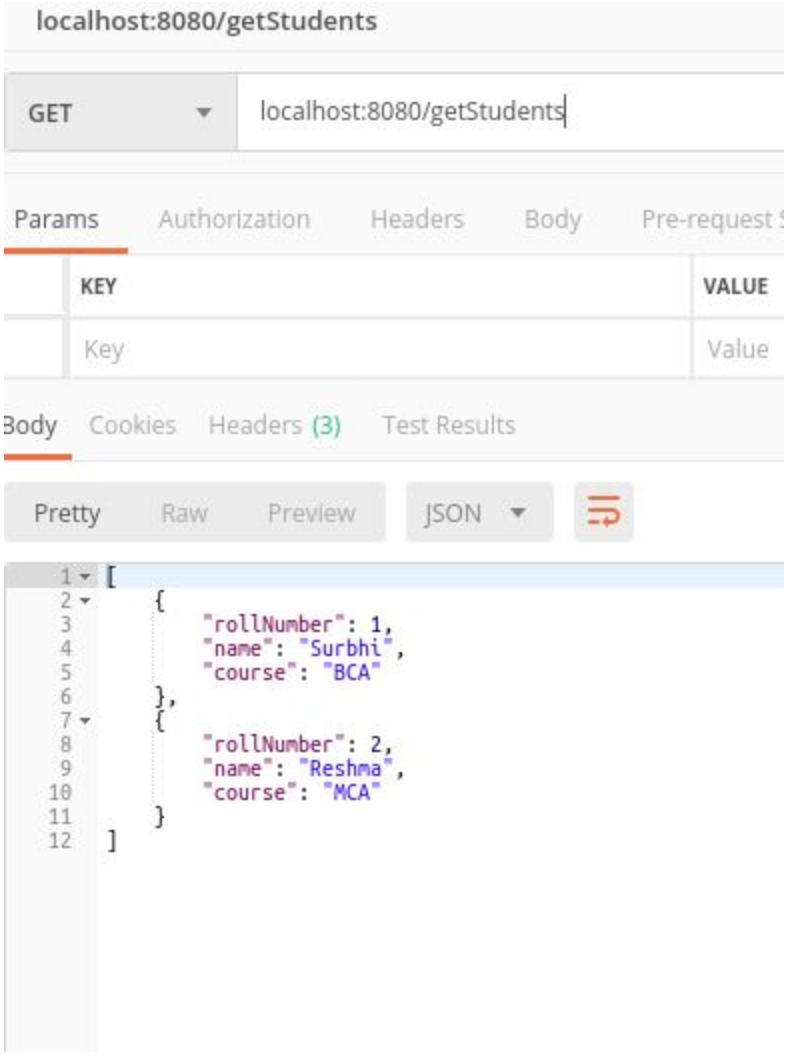
```
package com.spring.springBootDemo.controllers;

import com.spring.springBootDemo.entities.Student;
import com.spring.springBootDemo.services.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
```

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

import java.util.List;

@RestController
public class StudentController {
  @Autowired
  StudentService studentService;
  @RequestMapping("/getStudents")
  public List<Student> getStudentsList()
  {
     return studentService.getStudentsList();
  }
}
```

StudentService.java

```
package com.spring.springBootDemo.services;

import com.spring.springBootDemo.entities.Student;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.List;

@Service
public class StudentService {
  public List<Student> getStudentsList()
  {
     return Arrays.asList(
           new Student(1,"Surbhi","BCA"),
           new Student(2,"Reshma","MCA")
     );
  }
}
```

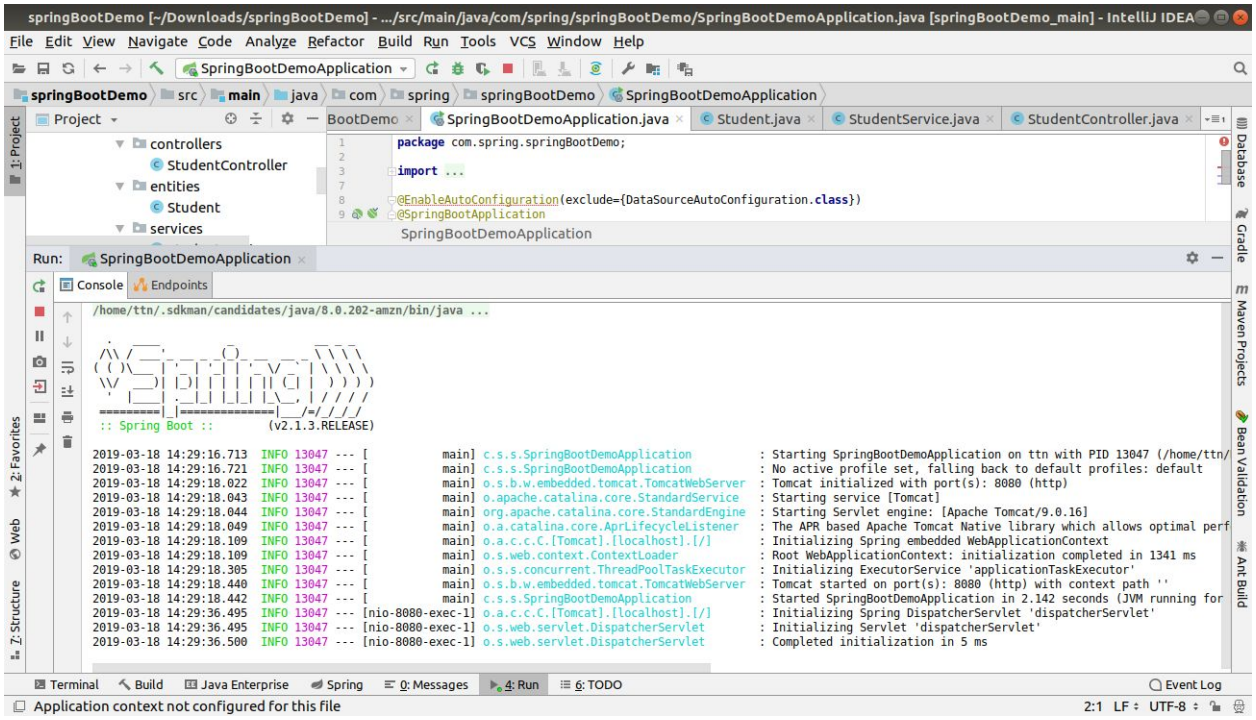
Output


localhost:8080/getStudents
GET
localhost:8080/getStudents
Params
Authorization
Headers
Body
Pre-request
KEY
VALUE
Key
Value
Body
Cookies
Headers (3)
Test Results
Pretty
Raw
Preview
JSON
[
{
"rollNumber": 1,
"name": "Surbhi",
"course": "BCA"
},
{
"rollNumber": 2,
"name": "Reshma",
"course": "MCA"
}
]

Q2. Run Spring Boot Application with all the three ways

Solution

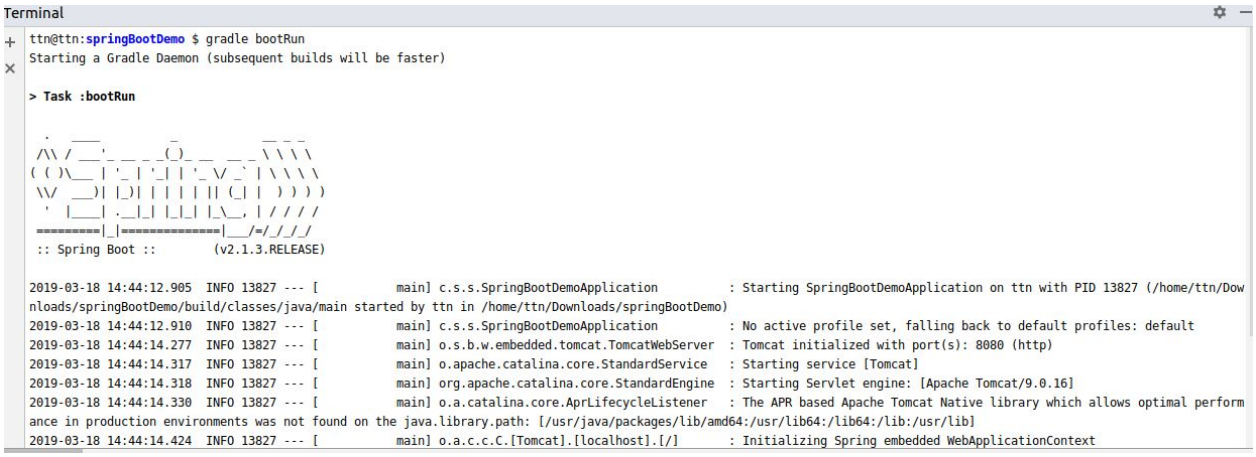
1)Right click on the class marked with @SpringBootApplication containing main method-->Click on run SpringBootDemoApplication.main()



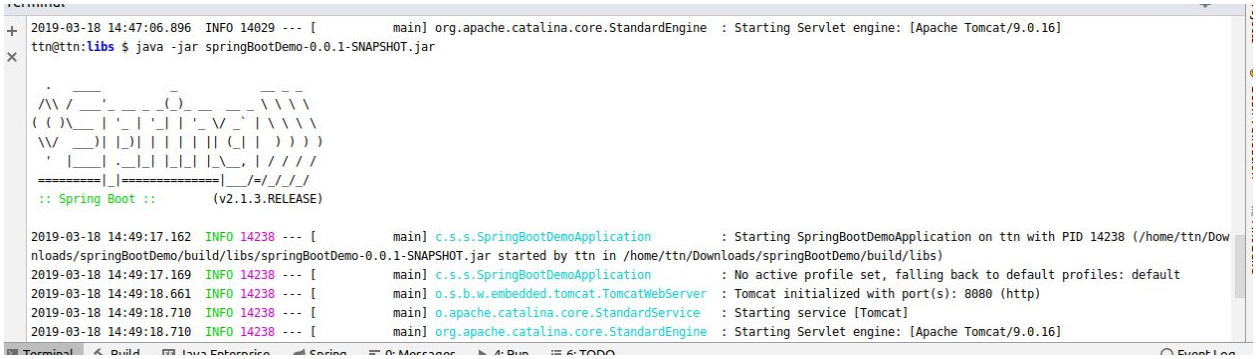
2)Execute bootRun gradletask

Open terminal→ gradle bootRun'

Or in the gradle tool window double click on bootRun task



3)Execute bootJar task from gradle window
Go to terminal
Cd build/libs
Run java -jar <jarname>



Q3. Create Bean User containing two field username and password with Spring Context File

Solution

Users.java

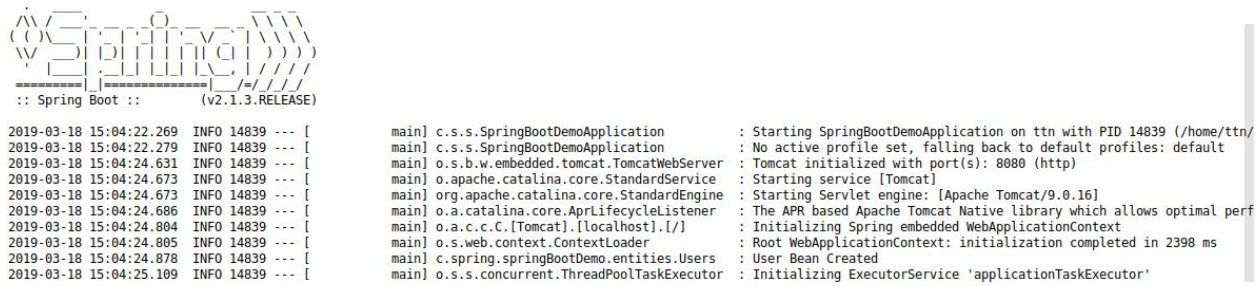
```
package com.spring.springBootDemo.entities;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

public class Users {
  private String username;
  private String password;

  Logger logger= LoggerFactory.getLogger(Users.class);
  public Users() {
     logger.info("User Bean Created");
  }

  public String getUsername() {
     return username;
  }

  public void setUsername(String username) {
     this.username = username;
  }

  public String getPassword() {
     return password;
  }

  public void setPassword(String password) {
     this.password = password;
  }
}
```

SpringBootDemoApplication.java

```
package com.spring.springBootDemo;

import com.spring.springBootDemo.entities.Users;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.context.annotation.Bean;

@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})
@SpringBootApplication
public class SpringBootDemoApplication {
```

```
@Bean
 Users user()
 {
   Users user=new Users();
   user.setUsername("surbhi.garg@tothenew.com");
   user.setPassword("Hello");
   return user;
 }
 public static void main(String[] args) {
   SpringApplication.run(SpringBootDemoApplication.class, args);
 }

}
```

Output



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.3.RELEASE)

2019-03-18 15:04:22.269  INFO 14839 --- [           main] c.s.s.SpringBootDemoApplication          : Starting SpringBootDemoApplication on ttn with PID 14839 (/home/ttn/
2019-03-18 15:04:22.279  INFO 14839 --- [           main] c.s.s.SpringBootDemoApplication          : No active profile set, falling back to default profiles: default
2019-03-18 15:04:24.631  INFO 14839 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2019-03-18 15:04:24.673  INFO 14839 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-03-18 15:04:24.673  INFO 14839 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.16]
2019-03-18 15:04:24.686  INFO 14839 --- [           main] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which allows optimal perf
2019-03-18 15:04:24.804  INFO 14839 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-03-18 15:04:24.805  INFO 14839 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2398 ms
2019-03-18 15:04:24.878  INFO 14839 --- [           main] c.spring.springBootDemo.entities.Users   : User Bean Created
2019-03-18 15:04:25.109  INFO 14839 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
```

Q4.Create a Bean Database with two fields port and name and Access its values using application.properties

Solution

Application.properties

```
database.name="mysql"
database.port=3306
```

Database.java

```
package com.spring.springBootDemo.entities;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Database {
  @Value("${database.name}")
 private  String name;
  @Value("${database.port}")
private  int port;

  public String getName() {
     return name;
  }

  public int getPort() {
     return port;
  }
}
```

DatabaseService.java

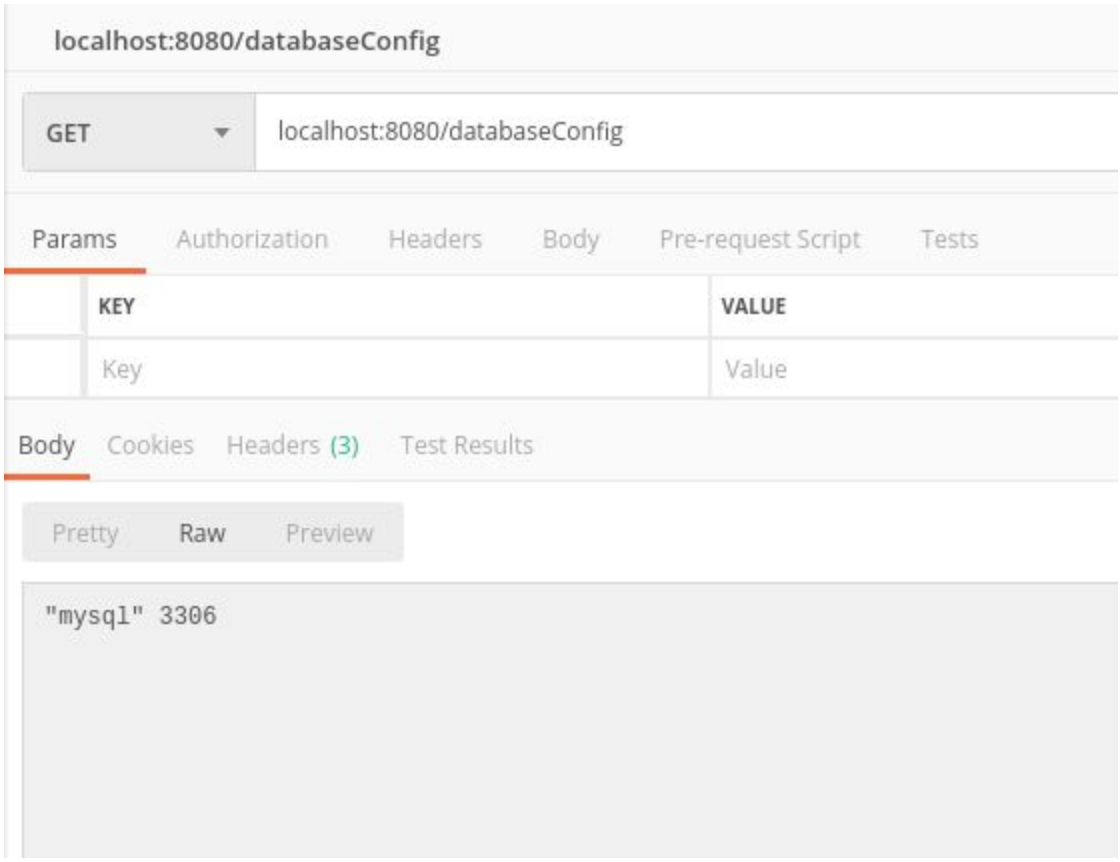
```
package com.spring.springBootDemo.services;

import com.spring.springBootDemo.entities.Database;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.xml.crypto.Data;
@Service
public class DatabaseService {
  @Autowired
  Database database;
  public String getDataBaseConfig()
  {
     return database.getName()+" "+database.getPort();
  }
}
```

DatabaseController.java

```
package com.spring.springBootDemo.controllers;
```

```
import com.spring.springBootDemo.entities.Database;
import com.spring.springBootDemo.services.DatabaseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DatabaseController {
  @Autowired
  DatabaseService databaseService;

  @RequestMapping("/databaseConfig")
  public String databaseConfig()
  {
     return databaseService.getDataBaseConfig();
  }


}
```

Output



Q5. Configure environment specfic values for Database Bean

Solution

application-prod.properties

```
database.port=8826
database.name=oracle
```

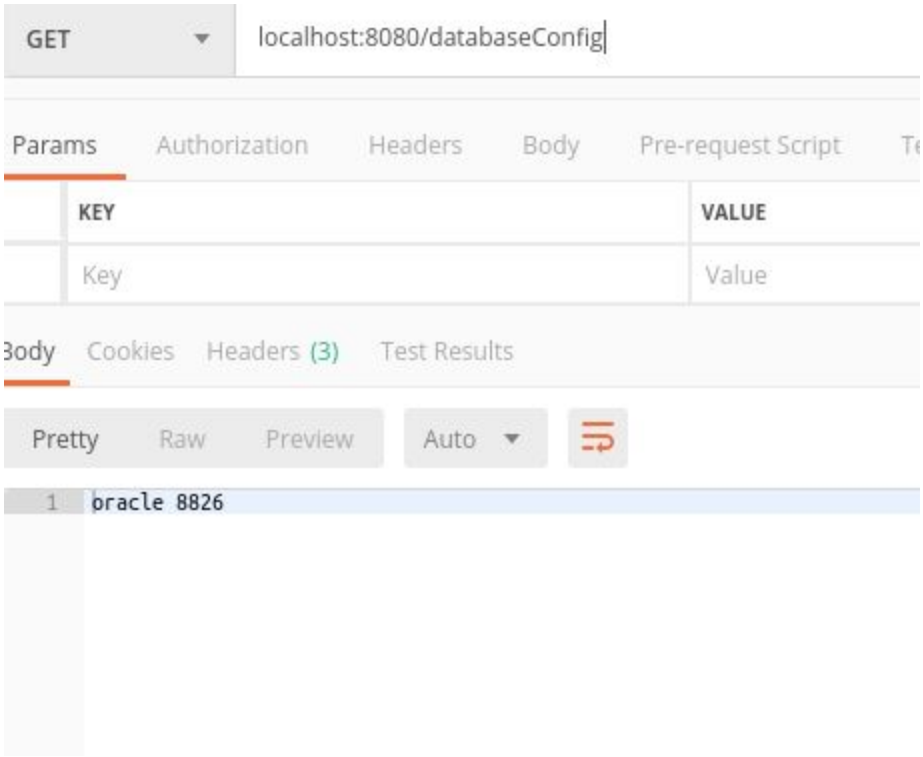
Database.java

Same as previous question

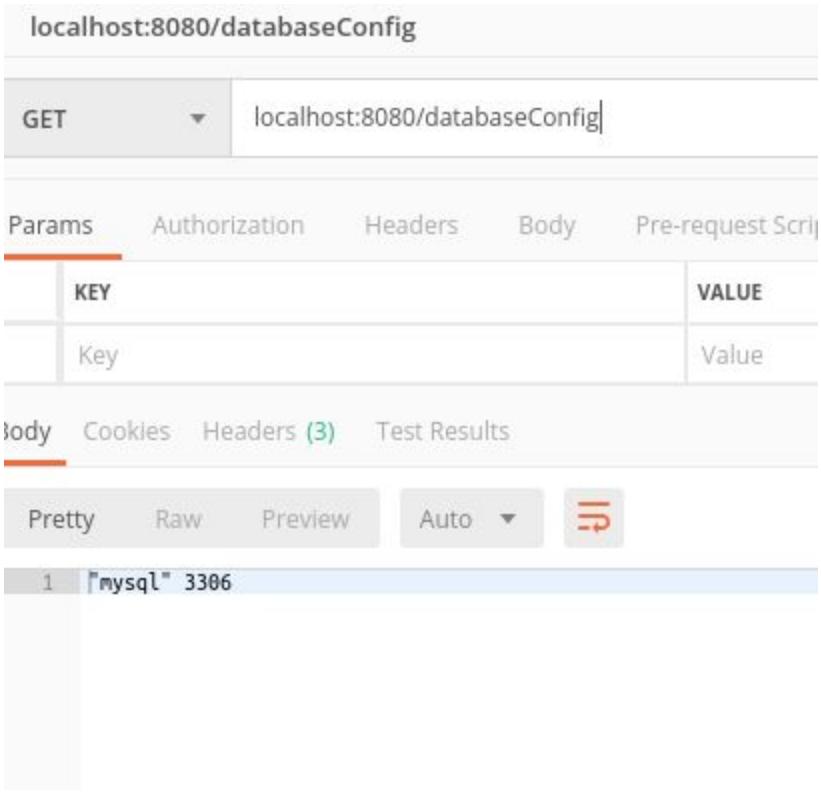
Output

1)Using

Java -jar <jarname> --spring.profiles.active=prod



2)Using java -jar <jarname>



Q6. Apply Logging whereever you feel is necessity

DatabaseController.java

```
package com.spring.springBootDemo.controllers;

import com.spring.springBootDemo.entities.Database;
import com.spring.springBootDemo.services.DatabaseService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;



@RestController
public class DatabaseController {
  @Autowired
  DatabaseService databaseService;

  Logger logger= LoggerFactory.getLogger(DatabaseController.class);

  @RequestMapping("/databaseConfig")
  public String databaseConfig()
  {
     //Question6
     logger.info("------Calling databaseService.getDataBaseConfig()-------");
     return databaseService.getDataBaseConfig();
  }


}
```

StudentController.java

```
package com.spring.springBootDemo.controllers;

import com.spring.springBootDemo.entities.Student;
import com.spring.springBootDemo.services.StudentService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class StudentController {
```

```
  @Autowired
  StudentService studentService;
  Logger logger= LoggerFactory.getLogger(StudentController.class);
  @RequestMapping("/getStudents")
  public List<Student> getStudentsList()
  {
     //Question6
     logger.info(".....Calling studentService.getStudentsList().....");
     return studentService.getStudentsList();
  }
}
```

DatabaseService.java

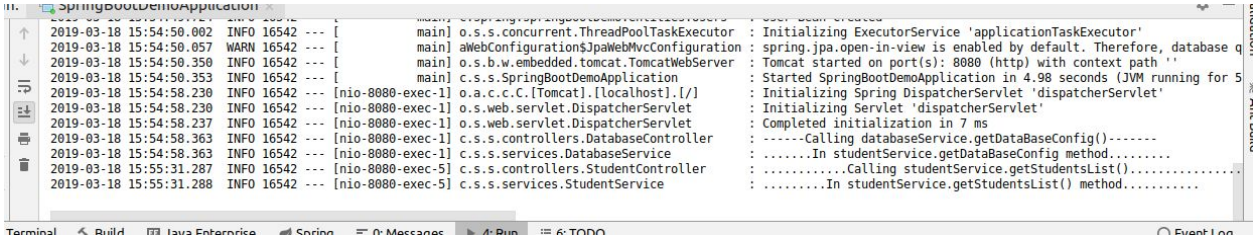
```
package com.spring.springBootDemo.services;

import com.spring.springBootDemo.entities.Database;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import sun.rmi.server.LoaderHandler;

@Service
public class DatabaseService {
  @Autowired
  Database database;
  Logger logger= LoggerFactory.getLogger(DatabaseService.class);
  public String getDataBaseConfig()
  {
     //Question6
     logger.info(".....In studentService.getDataBaseConfig method.....");
     return database.getName()+" "+database.getPort();
  }
}
```

StudentService.java

```
package com.spring.springBootDemo.services;

import com.spring.springBootDemo.entities.Student;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.List;

@Service
public class StudentService {
  Logger logger= LoggerFactory.getLogger(StudentService.class);
  public List<Student> getStudentsList()
  {
     //Question6
```

```
        logger.info(".....In studentService.getStudentsList() method.....");
        return Arrays.asList(
                new Student(1,"Surbhi","BCA"),
                new Student(2,"Reshma","MCA")
        );
    }
}
```

Output



Q7. Bootstrap Data for Student Domain

Solution

Student.java

```
package com.spring.springBootDemo.entities;

import javax.persistence.Entity;
import javax.persistence.Id;

//added annotations for question7
@Entity
public class Student {
  @Id
  private int rollNumber;
  private String name;
  private String course;

  public Student() {
  }

  public int getRollNumber() {
    return rollNumber;
  }

  public void setRollNumber(int rollNumber) {
    this.rollNumber = rollNumber;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getCourse() {
    return course;
  }

  public void setCourse(String course) {
    this.course = course;
  }

  public Student(int rollNumber, String name, String course) {
    this.rollNumber = rollNumber;
    this.name = name;
    this.course = course;
  }
}
```

StudentRepository.java

```
package com.spring.springBootDemo.repositories;

import com.spring.springBootDemo.entities.Student;
import org.springframework.data.repository.CrudRepository;

public interface StudentRepository extends CrudRepository<Student,Integer> {
}
```

Bootstrap.java

```
package com.spring.springBootDemo.events;

import com.spring.springBootDemo.repositories.StudentRepository;
import com.spring.springBootDemo.entities.Student;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.ContextRefreshedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

import java.util.Iterator;

@Component
public class BootStrap {
  @Autowired
  StudentRepository studentRepository;
  @EventListener(ContextRefreshedEvent.class)
  public void insertData()
  {
     Iterator<Student>studentIterator=studentRepository.findAll().iterator();
     if(!(studentIterator.hasNext()))
     {
        for(int i=0;i<5;i++)
        {
           Student student=new Student(i+1,"Student"+i,"MCA");
           studentRepository.save(student);
           System.out.println("student"+i+" created");
        }
     }
  }
}
```

Output

```
2019-03-18 16:18:21.753  INFO 17650 --- [           main] c.spring.springBootDemo.entities.Users   : User Bean Created
2019-03-18 16:18:22.031  INFO 17650 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-03-18 16:18:22.091  WARN 17650 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database q
2019-03-18 16:18:22.416  INFO 17650 --- [           main] o.h.h.i.QueryTranslatorFactoryInitiator  : HHH000397: Using ASTQueryTranslatorFactory
student0 created
student1 created
student2 created
student3 created
student4 created
2019-03-18 16:18:22.668  INFO 17650 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2019-03-18 16:18:22.670  INFO 17650 --- [           main] c.s.s.SpringBootDemoApplication          : Started SpringBootDemoApplication in 6.117 seconds (JVM running for
```

rminal | Build | Java Enterprise | Spring | 0: Messages | 4: Run | 6: TODO | Event Log

Terminal

File Edit View Search Terminal Help

```
mysql> select * from student;
+-------------+--------+----------+
| roll_number | course | name     |
+-------------+--------+----------+
|           1 | MCA    | Student0 |
|           2 | MCA    | Student1 |
|           3 | MCA    | Student2 |
|           4 | MCA    | Student3 |
|           5 | MCA    | Student4 |
+-------------+--------+----------+
5 rows in set (0.00 sec)

mysql>
```