HOSPITAL MANAGEMENT

(SPRING BOOT PROJECT)

INTRODUCTION:

Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

SYSTEM SOFTWARE USED:

Spring Boot Project:

There are multiple approaches to create Spring Boot project. We can use any of the following approach to create application.

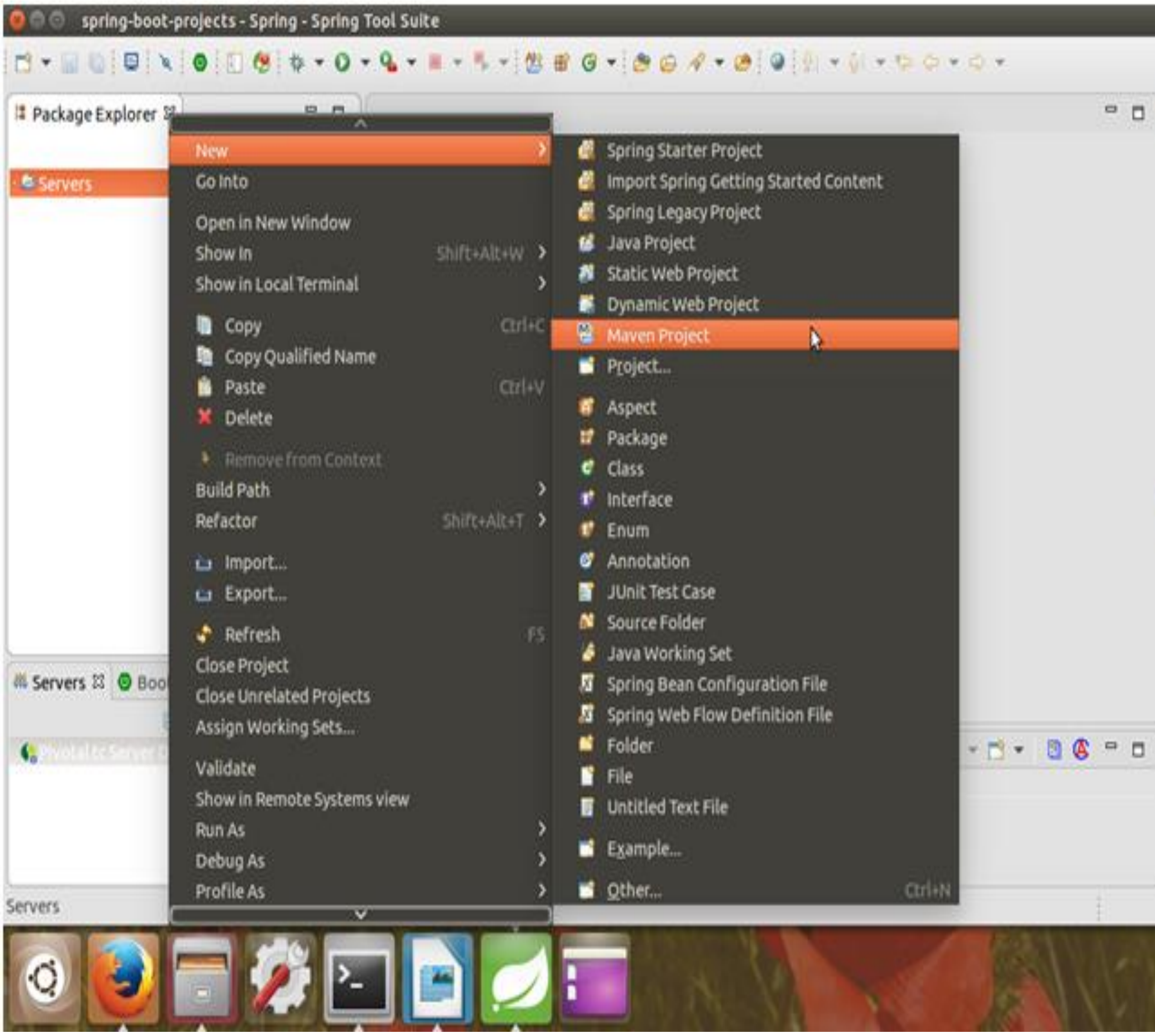
- Spring Maven Project
- Spring Starter Project Wizard
- Spring Initializer
- Spring Boot CLI

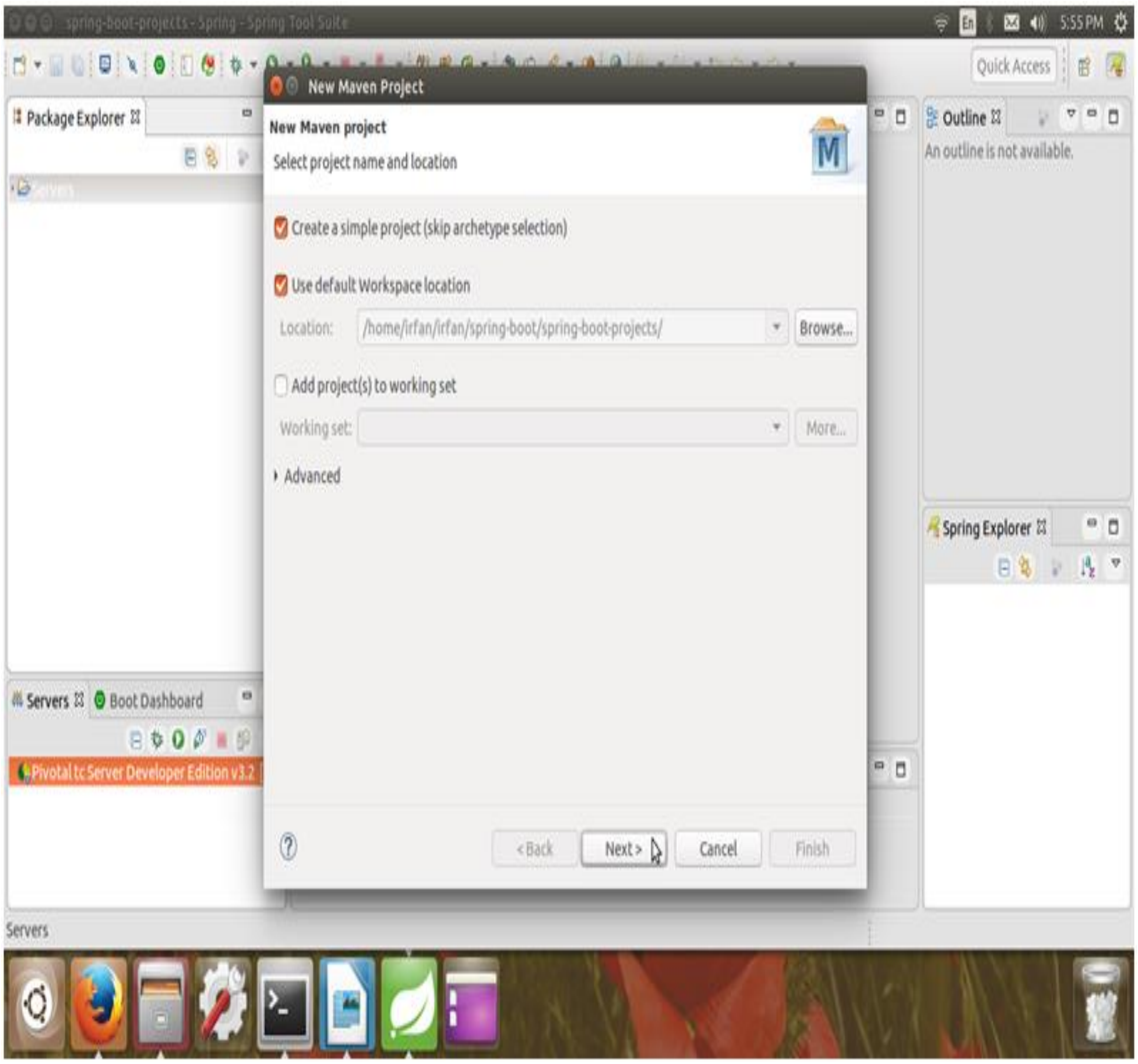
Here for all the example, we are using STS (Spring Toll Suite) IDE to create project. You can download this IDE from official site of Spring framework.

Spring Boot Maven Project

Creating Spring Boot project by creating maven project. It includes the following steps.

How to create an mavan project using spring tool suite with a following steps ,steps have to be followed while creating mavan project ,


spring-boot-projects - Spring - Spring Tool Suite
Package Explorer
Servers
New
Go Into
Open in New Window
Show In Shift+Alt+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete
Remove from Context
Build Path
Refactor Shift+Alt+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Validate
Show in Remote Systems view
Run As
Debug As
Profile As
Spring Starter Project
Import Spring Getting Started Content
Spring Legacy Project
Java Project
Static Web Project
Dynamic Web Project
Maven Project
Project...
Aspect
Package
Class
Interface
Enum
Annotation
JUnit Test Case
Source Folder
Java Working Set
Spring Bean Configuration File
Spring Web Flow Definition File
Folder
File
Untitled Text File
Example...
Other... Ctrl+N
Servers



spring-boot-projects - Spring - Spring Tool Suite
5:55 PM
Quick Access
Package Explorer
Servers
Boot Dashboard
Pivotal tc Server Developer Edition v3.2
Outline
An outline is not available.
Spring Explorer
Servers

New Maven Project

New Maven project

Select project name and location

[✓] Create a simple project (skip archetype selection)

[✓] Use default Workspace location

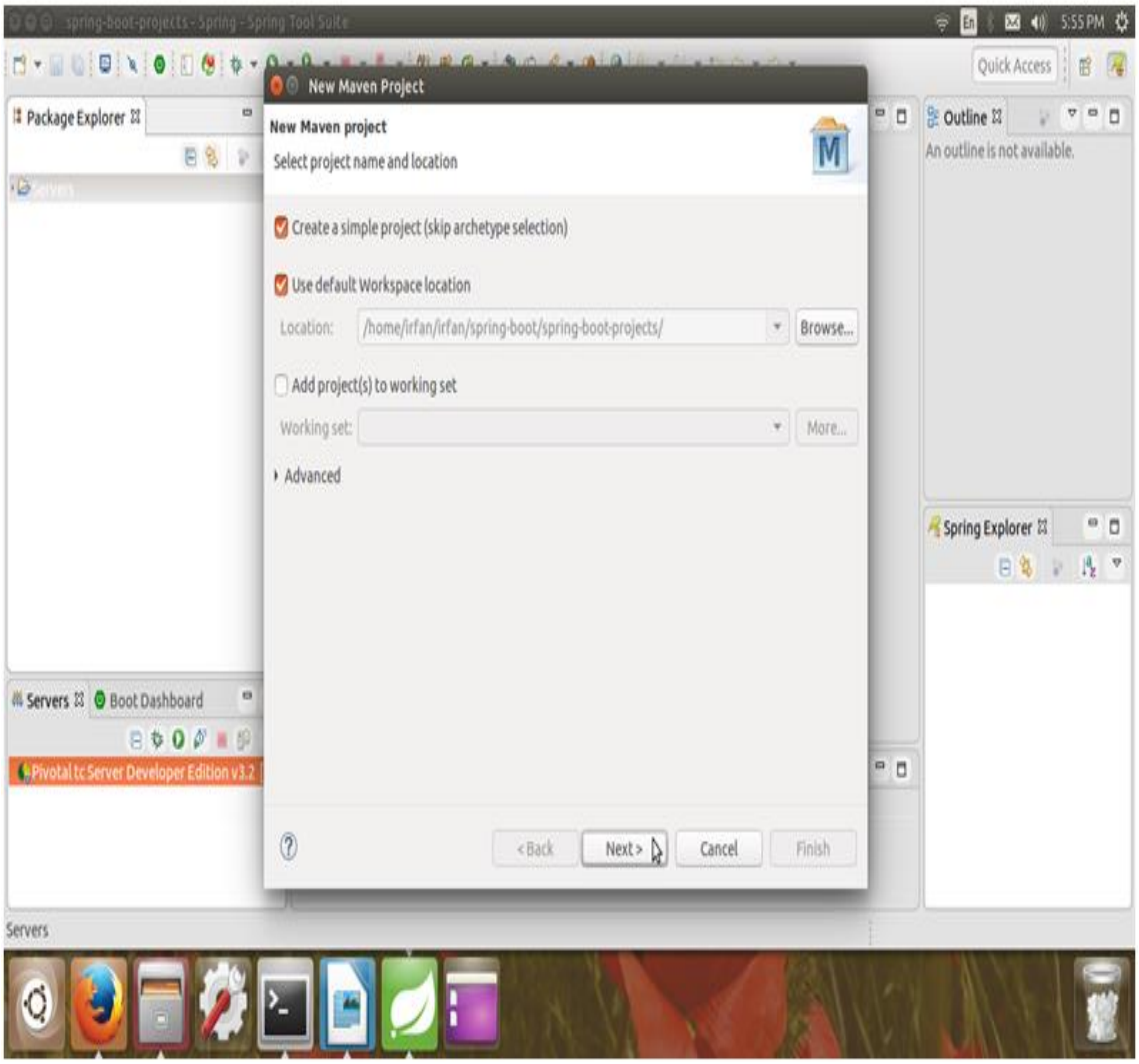
Location: /home/irfan/irfan/spring-boot/spring-boot-projects/ Browse...

[ ] Add project(s) to working set

Working set: More...

Advanced

< Back Next > Cancel Finish


New Maven project

Select project name and location

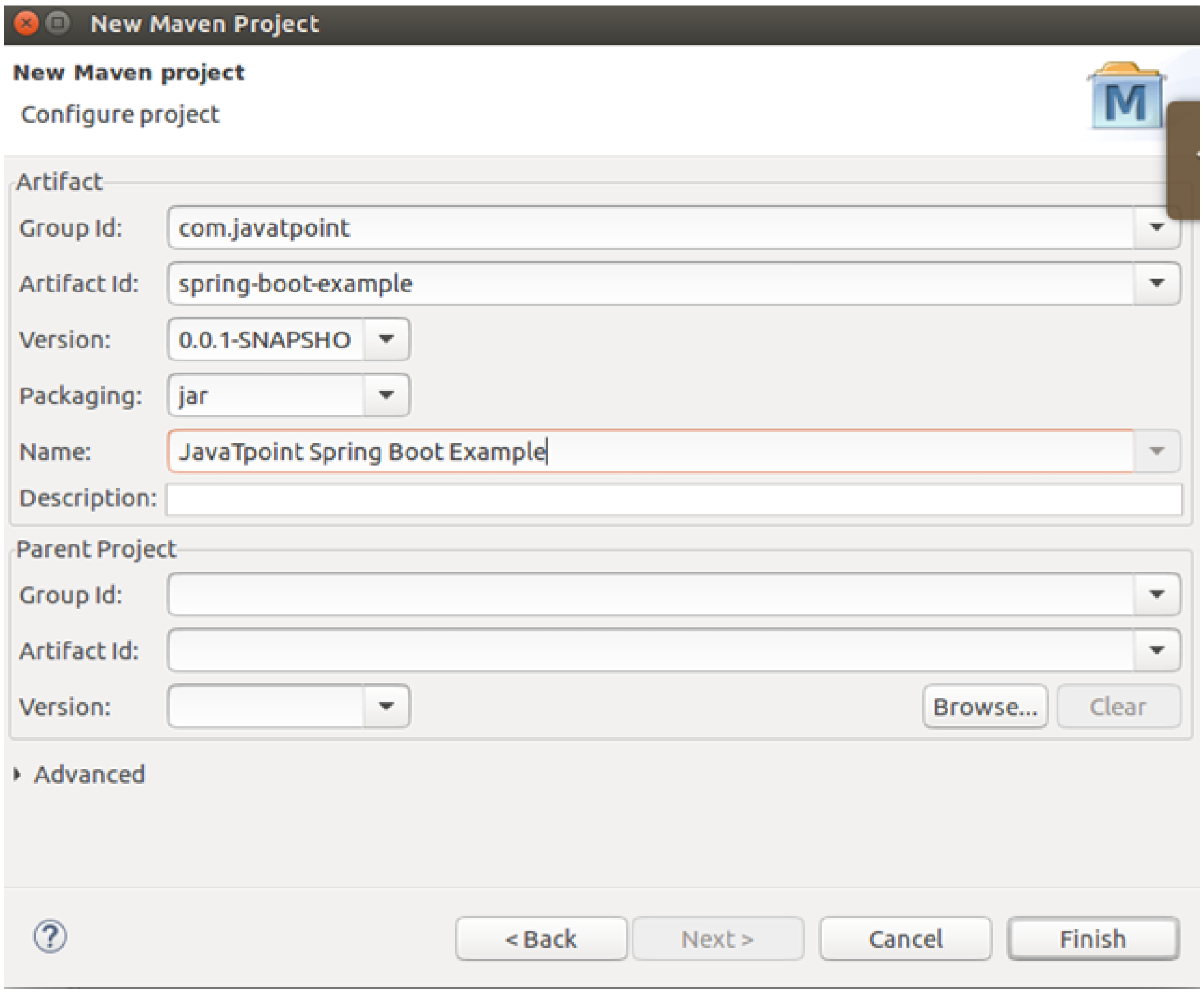
[✓] Create a simple project (skip archetype selection)

[✓] Use default Workspace location

Location: /home/irfan/irfan/spring-boot/spring-boot-projects/

[ ] Add project(s) to working set

Working set:


New Maven Project

New Maven project

Configure project

Artifact

Group Id: com.javatpoint

Artifact Id: spring-boot-example

Version: 0.0.1-SNAPSHO

Packaging: jar

Name: JavaTpoint Spring Boot Example

Description:

Parent Project

Group Id:

Artifact Id:

Version:

This Maven project has a pom.xml file which contains the following default configuration.

// pom.xml

```
<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.javatpoint</groupId>
<artifactId>spring-boot-example</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

```
6.  <name>JavaTpoint Spring Boot Example</name>
7.  </project>
```

Spring Boot Annotations:

AUTOWIRED:

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

@Autowired on Setter Methods

You can use @Autowired annotation on setter methods to get rid of the <property> element in XML configuration file. When Spring finds an @Autowired annotation used with setter methods, it tries to perform byType autowiring on the method.

Example

```
@Component

public class Customer

{

private Person person;

@Autowired
public Customer(Person person)
{
this.person=person;
}
}
```

POSTMAPPING:

The @PostMapping is specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST).

The @PostMapping annotated methods in the @Controller annotated classes handle the HTTP POST requests matched with given URI expression.

@PostMapping Example:

```
@PostMapping(path = "users",
consumes = MediaType.APPLICATION_JSON_VALUE,
produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<User> create(@RequestBody User newUser) {
User user = userService.save(newUser);
if (user == null) {
throw new ServerException();
} else {
return new ResponseEntity<>(user, HttpStatus.CREATED);
}

}
```

GETMAPPING:

The @GetMapping annotation is a specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

The @GetMapping annotated methods in the @Controller annotated classes handle the HTTP GET requests matched with given URI expression.

@GetMapping Example:

```
@RestController

public class UserController {


@Autowired

UserService userService;


@GetMapping("users")

public ResponseEntity<List<User>> getAll() {

return new ResponseEntity<>(userService.getAll(), HttpStatus.OK);

}


@GetMapping("users/{id}")
```

```
public ResponseEntity<User> getById(@PathVariable long id) {
Optional<User> user = userService.getById(id);
if (user.isPresent()) {
return new ResponseEntity<>(user.get(), HttpStatus.OK);
} else {
throw new RecordNotFoundException();
}
```

DELETEMAPPING:

The DELETE HTTP method is used to delete the resource and @DeleteMapping annotation for mapping HTTP DELETE requests onto specific handler methods.

Specifically, @DeleteMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.DELETE).

Example:

```
@DeleteMapping("/employees/{id}")
public Map<String, Boolean> deleteEmployee(@PathVariable(value = "id") Long employeeId)
throws ResourceNotFoundException {
Employee employee = employeeRepository.findById(employeeId).orElseThrow(() ->
```

ResourceNotFoundException("Employee not found for this id :: " + employeeId));

employeeRepository.delete(employee);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return response;

}

PUTMAPPING

@PutMapping: It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates It is used instead of using: @RequestMapping(method = RequestMethod.PUT)

Example:

@PutMapping("/employees/{id}")

public Map<String, Boolean> deleteEmployee(@PathVariable(value = "id") Long employeeId)

throws ResourceNotFoundException {

Employee employee = employeeRepository.findById(employeeId)

.orElseThrow(() -> new ResourceNotFoundException("Employee not found for this id :: " + employeeId));

employeeRepository.delete(employee);

Map<String, Boolean> response = new HashMap<>();

response.put("deleted", Boolean.TRUE);

return response;

OVERRIDE:

The @Override annotation is a standard Java annotation that was first introduced in Java 1.5. The @Override annotation denotes that the child class method overrides the base class method.

For two reasons, the @Override annotation is useful.

If the annotated method does not actually override anything, the compiler issues a warning.

It can help to make the source code more readable.

id auto generated

Why we use @Override annotation:

Because of the following two advantages, using the @Override annotation when overriding a method is considered a best practice for coding in Java:

1) You'll get a compile-time error if the programmer makes a mistake while overriding, such as using the wrong method name or parameter types. Because you are informing the compiler that you are overriding this method by using this annotation. If you don't use the annotation, the sub-class method will be treated as a new method in the subclass (rather than the overriding method).

SERVICE: @Service annotation is used in your service layer and annotates classes that perform service tasks, often you don't use it but in many case you use this annotation to represent a best practice. For example, you could directly call a DAO class to persist an object to your database but this is horrible.

USED METHODOLOGY:

Get the patient details using GET METHOD in postman

GET Request in Postman

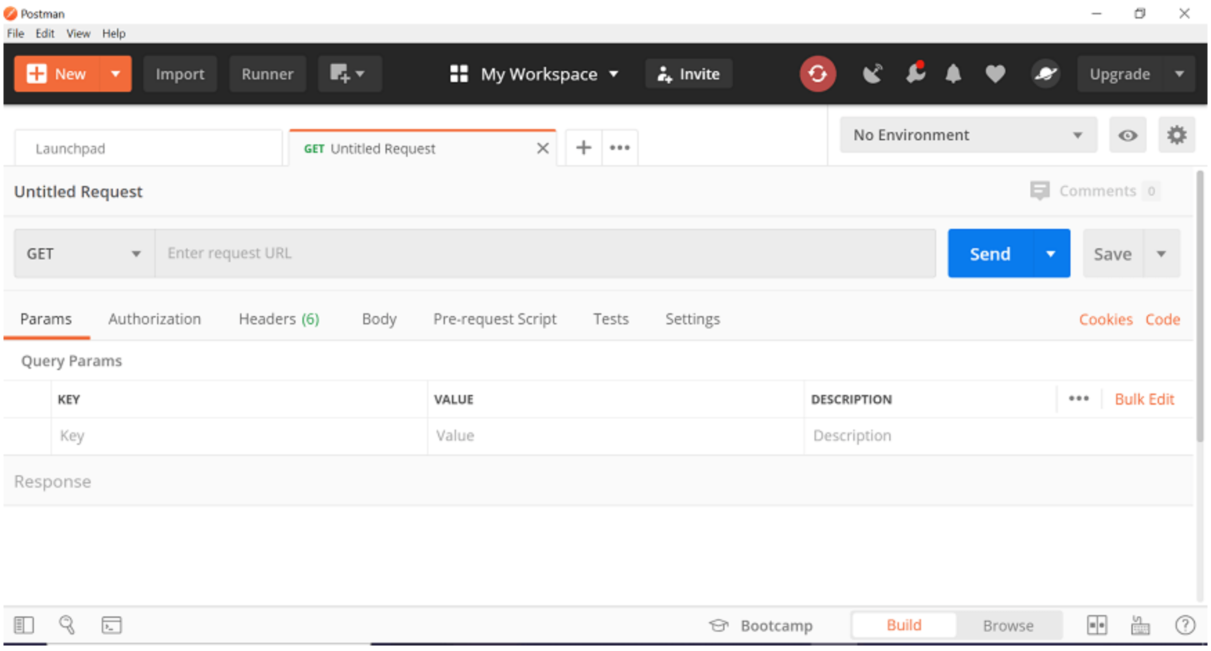
Since now we know how to create the request in Postman, it's time to work on GET request. A GET request gets the information from the server. When you make the GET request on the server, then the server responds to the request.

GET request will not affect any data on the server. Means, there is no creation, updation, addition, or deletion of data on the server when you are making a GET request.

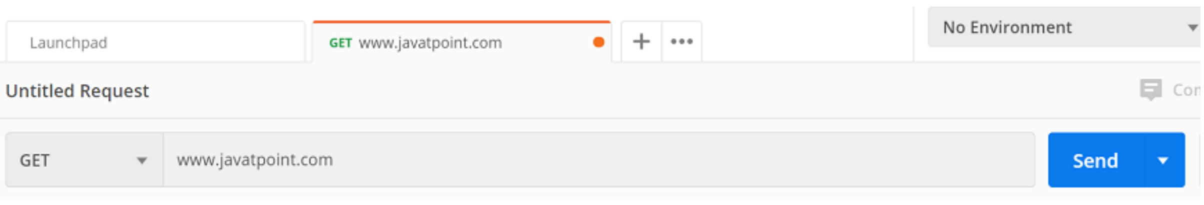
GET request contains all information inside the URL, and because of that, some people do not prefer to use GET request while they are sending some confidential data such as passwords. For example, if you search anything on Google, you actually using a GET request because there is no sensitive information, and you are just requesting the page. You can try to search for something on Google; you will get the same search string in the URL.

To create the first GET request in Postman, follow the following steps:

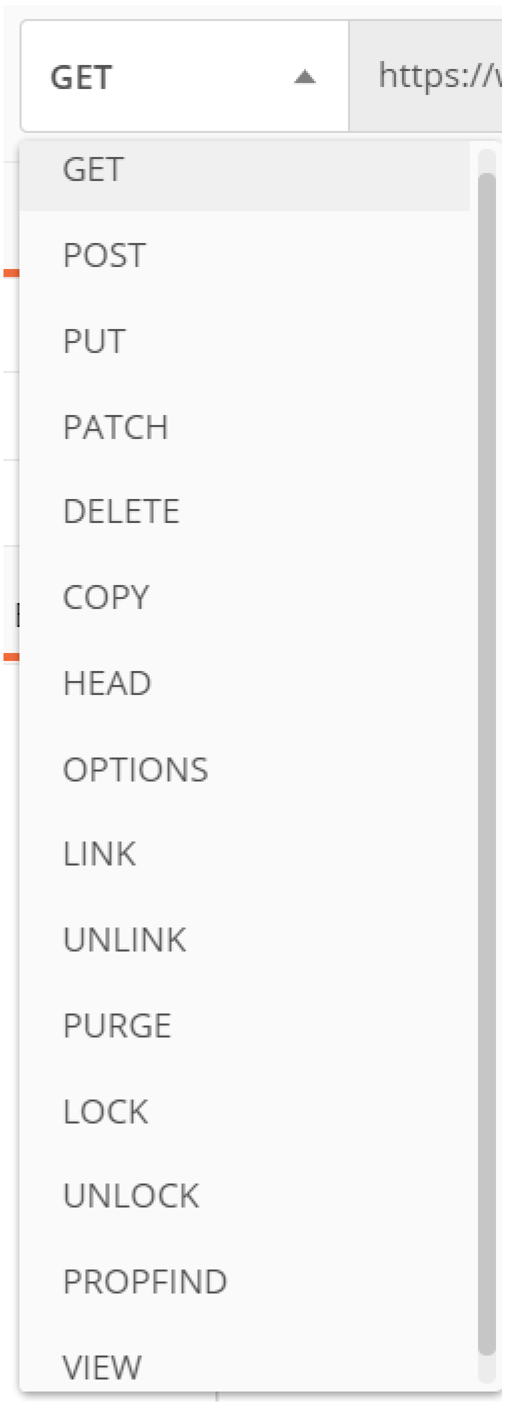
Step 1: Create a request. To create a new request, open a new tab, click from the **+** plus button.



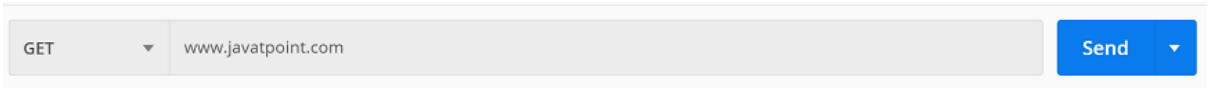
Step 2: Enter the URL in the address bar. We will use ,



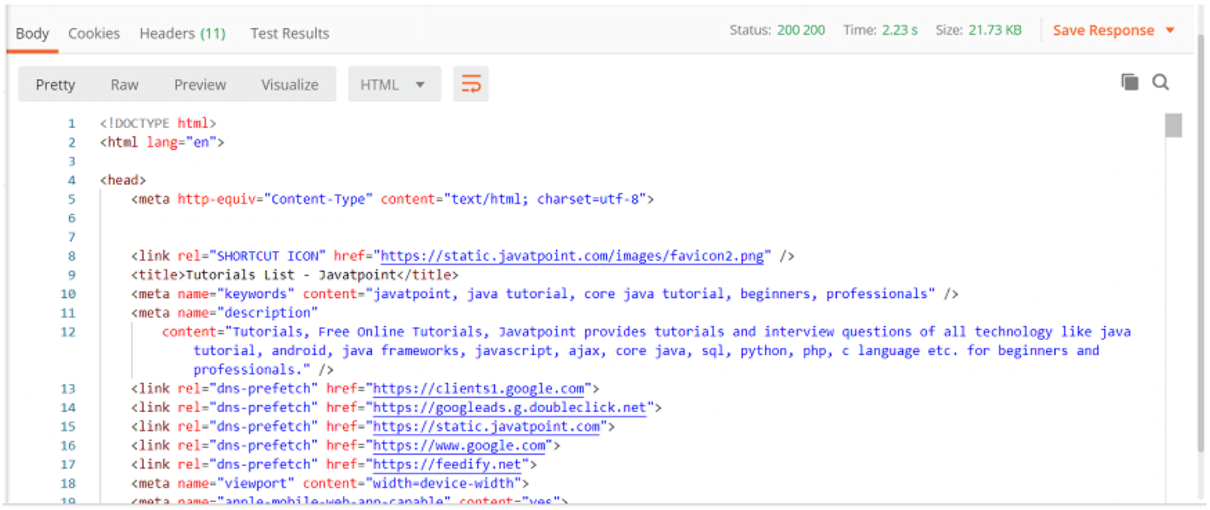
Step 3: Now, select the GET request from the list of request methods.



Step 4: Now press send.



Step 5: Once you press the send button, you will get the response from the server. Make sure you have a proper internet connection; otherwise, you will not get a response.

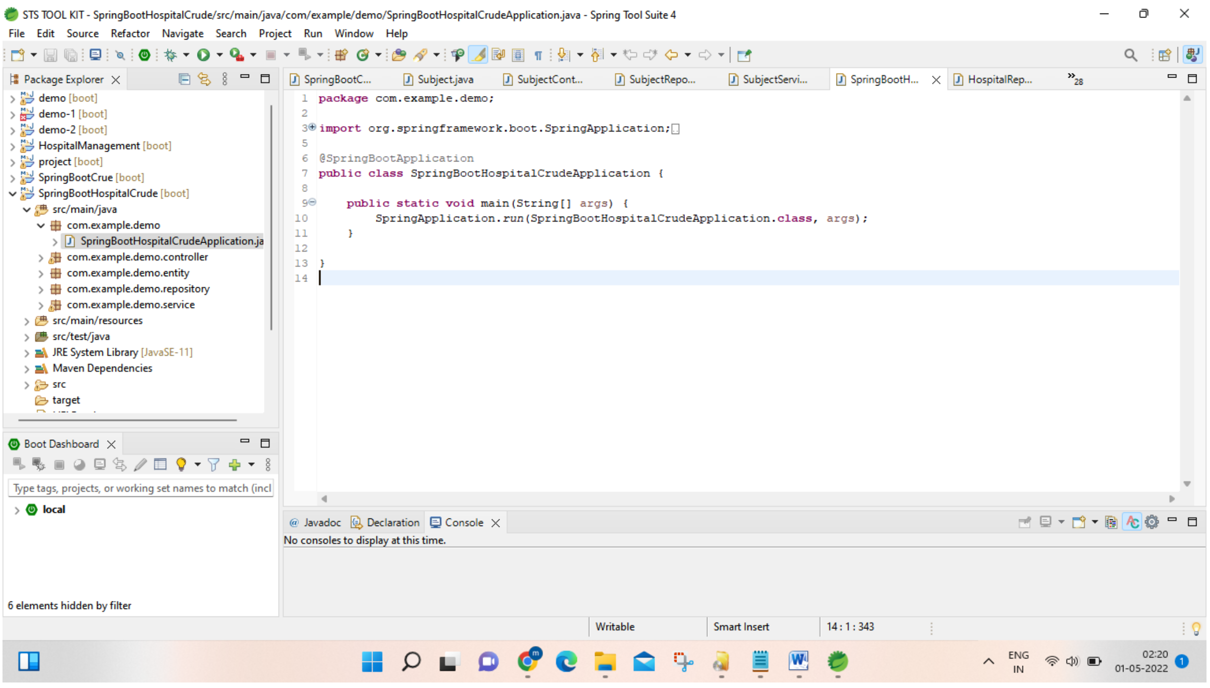


Body Cookies Headers (11) Test Results Status: 200 200 Time: 2.23 s Size: 21.73 KB Save Response

Pretty Raw Preview Visualize HTML

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">


    <link rel="SHORTCUT ICON" href="https://static.javatpoint.com/images/favicon2.png" />
    <title>Tutorials List - Javatpoint</title>
    <meta name="keywords" content="javatpoint, java tutorial, core java tutorial, beginners, professionals" />
    <meta name="description"
        content="Tutorials, Free Online Tutorials, Javatpoint provides tutorials and interview questions of all technology like java
            tutorial, android, java frameworks, javascript, ajax, core java, sql, python, php, c language etc. for beginners and
            professionals." />
    <link rel="dns-prefetch" href="https://clients1.google.com">
    <link rel="dns-prefetch" href="https://googleads.g.doubleclick.net">
    <link rel="dns-prefetch" href="https://static.javatpoint.com">
    <link rel="dns-prefetch" href="https://www.google.com">
    <link rel="dns-prefetch" href="https://feedify.net">
    <meta name="viewport" content="width=device-width">
    <meta name="apple-mobile-web-app-capable" content="yes">
```

SPRING PROJECT CLASSES WITH PACKAGE ALONG WITH SCREENSHOTS



STS TOOL KIT - SpringBootHospitalCrude/src/main/java/com/example/demo/SpringBootHospitalCrudeApplication.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

```
> demo [boot]
> demo-1 [boot]
> demo-2 [boot]
> HospitalManagement [boot]
> project [boot]
> SpringBootCrue [boot]
v SpringBootHospitalCrude [boot]
  v src/main/java
    v com.example.demo
      > SpringBootHospitalCrudeApplication.ja
    > com.example.demo.controller
    > com.example.demo.entity
    > com.example.demo.repository
    > com.example.demo.service
  > src/main/resources
  > src/test/java
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  > src
    target
```

SpringBootC... Subject.java SubjectCont... SubjectRepo... SubjectServi... SpringBootH... HospitalRep...

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SpringBootHospitalCrudeApplication {

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

}
```

Boot Dashboard

Type tags, projects, or working set names to match (incl

> local

6 elements hidden by filter

Javadoc Declaration Console

No consoles to display at this time.

Writable Smart Insert 14 : 1 : 343

ENG IN 02:20 01-05-2022

Main class with package demo ,

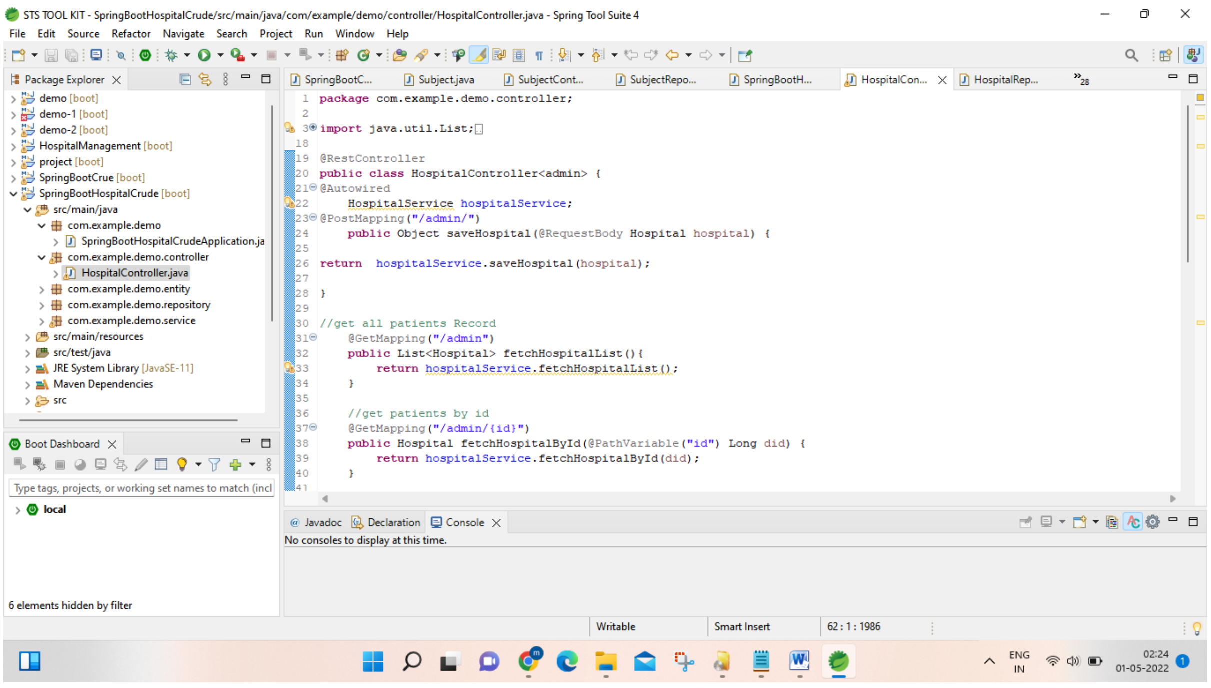
Java Main Method

public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method static by using the keyword static. We should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the main() method.

void: In Java, every method has the return type. Void keyword acknowledges the compiler that main() method does not return any value.

controller with class name HospitalController

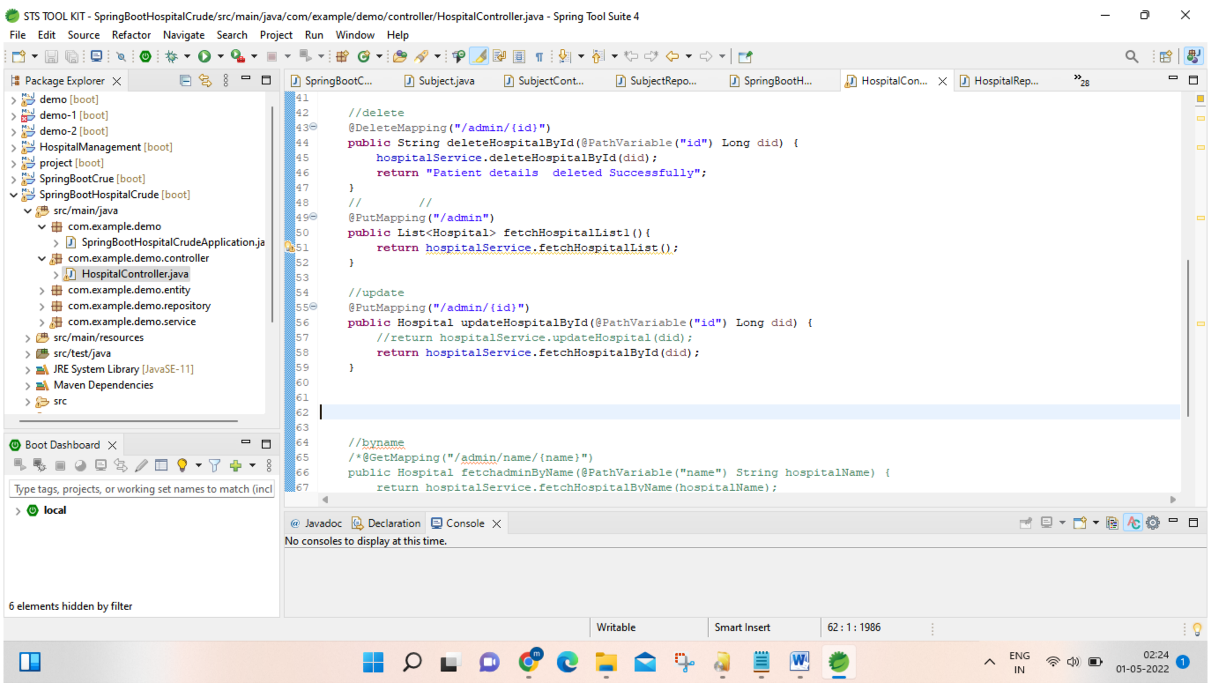


STS TOOL KIT - SpringBootHospitalCrude/src/main/java/com/example/demo/controller/HospitalController.java - Spring Tool Suite 4

```
package com.example.demo.controller;

import java.util.List;

@RestController
public class HospitalController<admin> {
@Autowired
    HospitalService hospitalService;
@PostMapping("/admin/")
    public Object saveHospital(@RequestBody Hospital hospital) {

return  hospitalService.saveHospital(hospital);

}

//get all patients Record
    @GetMapping("/admin")
    public List<Hospital> fetchHospitalList(){
        return hospitalService.fetchHospitalList();
    }

    //get patients by id
    @GetMapping("/admin/{id}")
    public Hospital fetchHospitalById(@PathVariable("id") Long did) {
        return hospitalService.fetchHospitalById(did);
    }
```

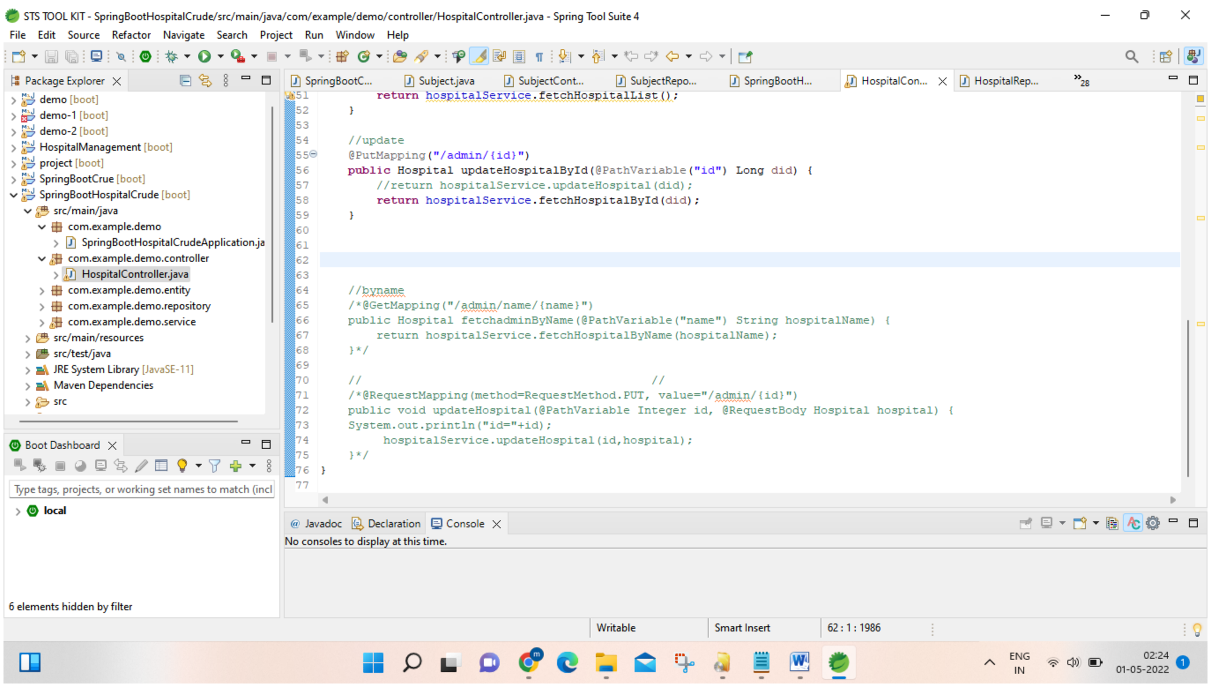
No consoles to display at this time.

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
demo [boot]
demo-1 [boot]
demo-2 [boot]
HospitalManagement [boot]
project [boot]
SpringBootCrue [boot]
SpringBootHospitalCrude [boot]
src/main/java
com.example.demo
SpringBootHospitalCrudeApplication.ja
com.example.demo.controller
HospitalController.java
com.example.demo.entity
com.example.demo.repository
com.example.demo.service
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
Boot Dashboard
Type tags, projects, or working set names to match (incl
local
6 elements hidden by filter
SpringBootC... Subject.java SubjectCont... SubjectRepo... SpringBootH... HospitalCon... HospitalRep...
Javadoc Declaration Console
Writable
Smart Insert
62 : 1 : 1986
ENG IN
02:24
01-05-2022



```java

    //delete
    @DeleteMapping("/admin/{id}")
    public String deleteHospitalById(@PathVariable("id") Long did) {
        hospitalService.deleteHospitalById(did);
        return "Patient details  deleted Successfully";
    }
    //        //
    @PutMapping("/admin")
    public List<Hospital> fetchHospitalList1(){
        return hospitalService.fetchHospitalList();
    }

    //update
    @PutMapping("/admin/{id}")
    public Hospital updateHospitalById(@PathVariable("id") Long did) {
        //return hospitalService.updateHospital(did);
        return hospitalService.fetchHospitalById(did);
    }




    //byname
    /*@GetMapping("/admin/name/{name}")
    public Hospital fetchadminByName(@PathVariable("name") String hospitalName) {
        return hospitalService.fetchHospitalByName(hospitalName);
```

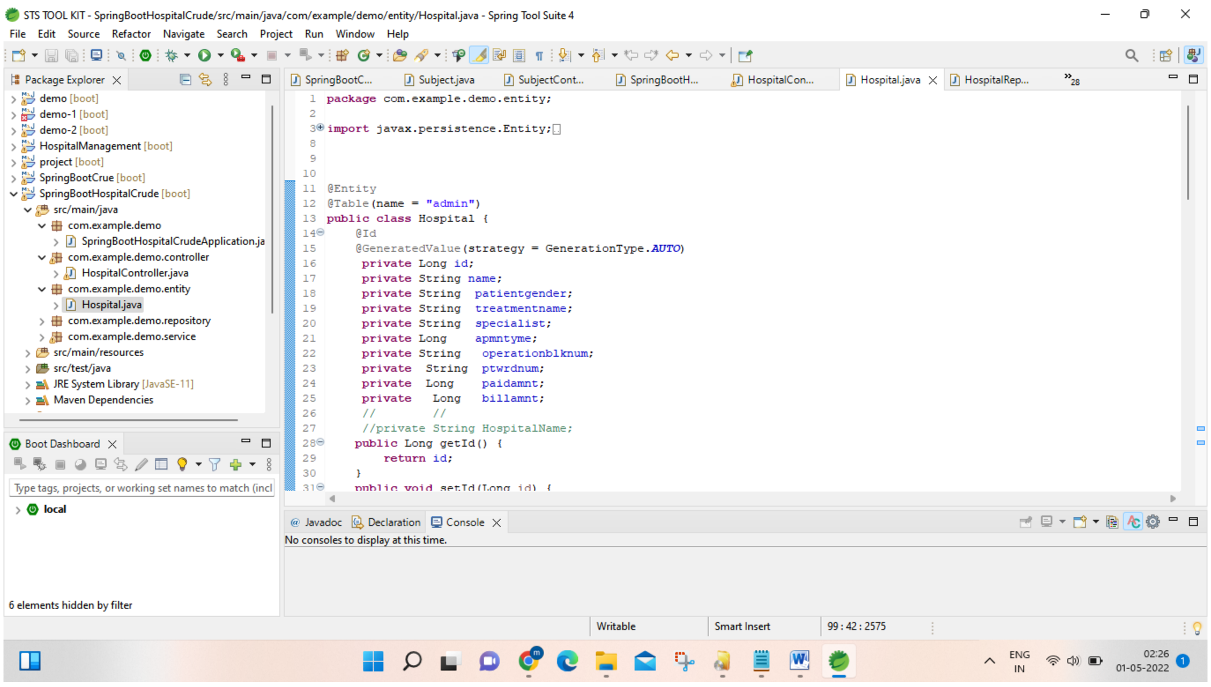
No consoles to display at this time.



```java
        return hospitalService.fetchHospitalList();
    }

    //update
    @PutMapping("/admin/{id}")
    public Hospital updateHospitalById(@PathVariable("id") Long did) {
        //return hospitalService.updateHospital(did);
        return hospitalService.fetchHospitalById(did);
    }




    //byname
    /*@GetMapping("/admin/name/{name}")
    public Hospital fetchadminByName(@PathVariable("name") String hospitalName) {
        return hospitalService.fetchHospitalByName(hospitalName);
    }*/

    //                                          //
    /*@RequestMapping(method=RequestMethod.PUT, value="/admin/{id}")
    public void updateHospital(@PathVariable Integer id, @RequestBody Hospital hospital) {
    System.out.println("id="+id);
         hospitalService.updateHospital(id,hospital);
    }*/
}

```

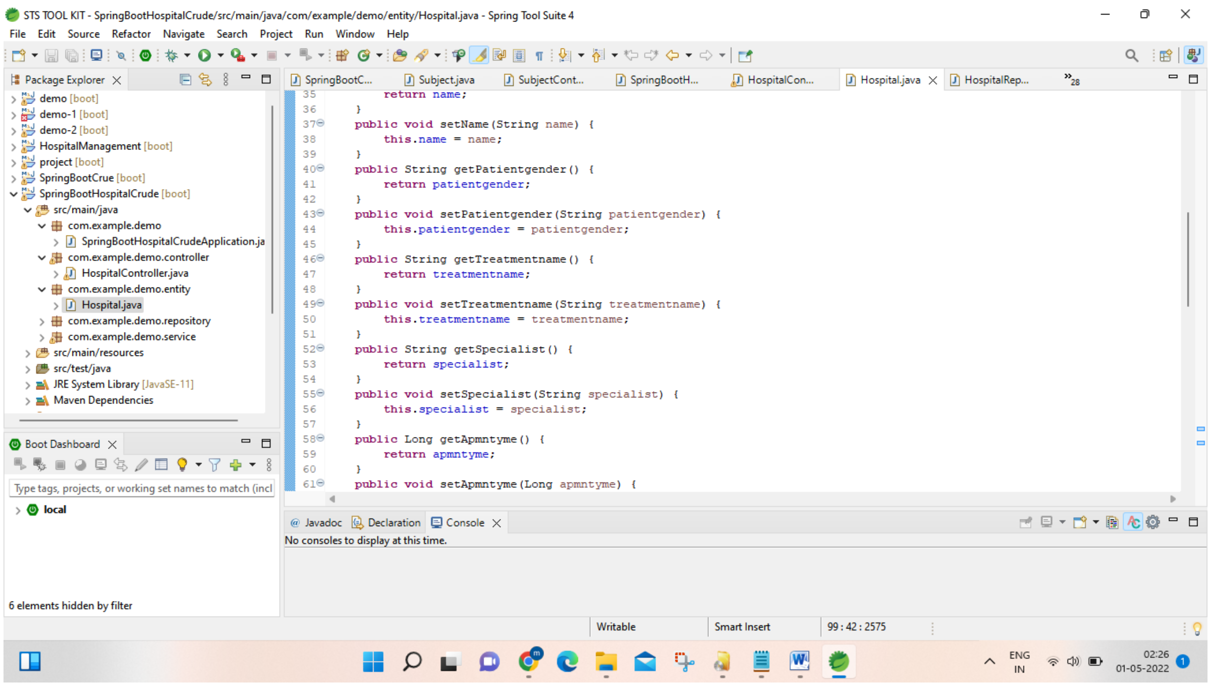
No consoles to display at this time.

entity with class name Hospital:



```java
package com.example.demo.entity;

import javax.persistence.Entity;



@Entity
@Table(name = "admin")
public class Hospital {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
     private Long id;
     private String name;
     private String  patientgender;
     private String  treatmentname;
     private String  specialist;
     private Long    apmntyme;
     private String   operationblknum;
     private  String  ptwrdnum;
     private  Long    paidamnt;
     private   Long   billamnt;
     //        //
     //private String HospitalName;
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
```

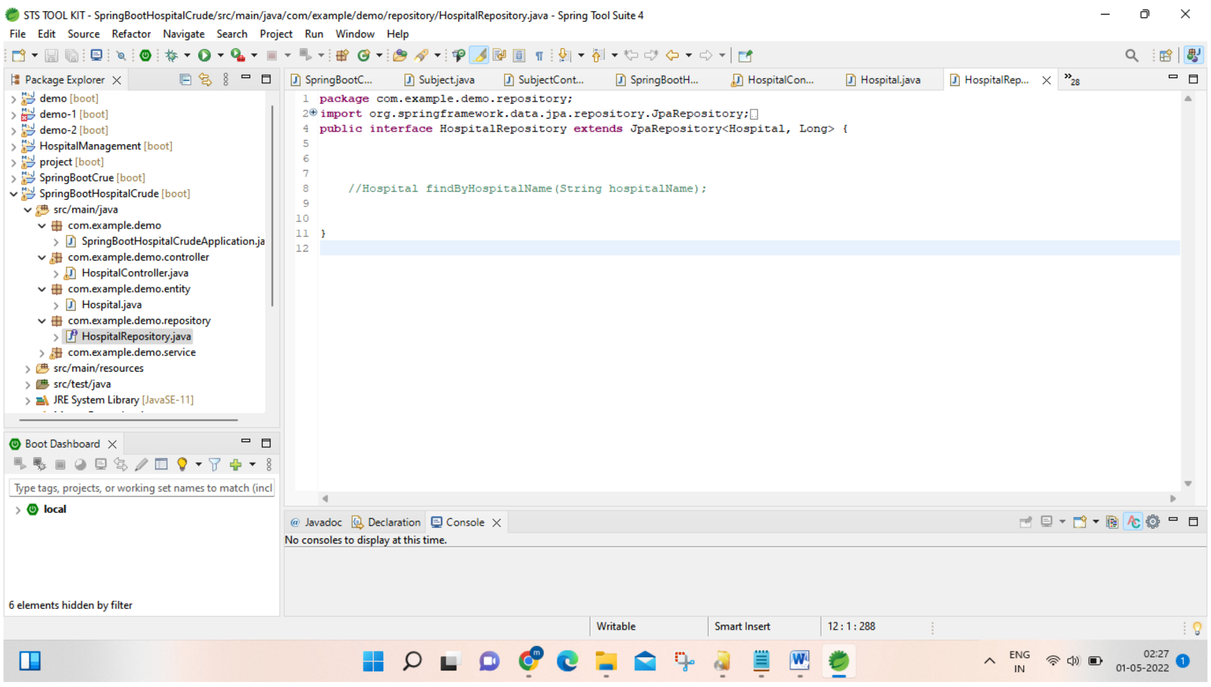
No consoles to display at this time.



```java
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPatientgender() {
        return patientgender;
    }
    public void setPatientgender(String patientgender) {
        this.patientgender = patientgender;
    }
    public String getTreatmentname() {
        return treatmentname;
    }
    public void setTreatmentname(String treatmentname) {
        this.treatmentname = treatmentname;
    }
    public String getSpecialist() {
        return specialist;
    }
    public void setSpecialist(String specialist) {
        this.specialist = specialist;
    }
    public Long getApmntyme() {
        return apmntyme;
    }
    public void setApmntyme(Long apmntyme) {
```

No consoles to display at this time.

repository with class name HospitalRepository

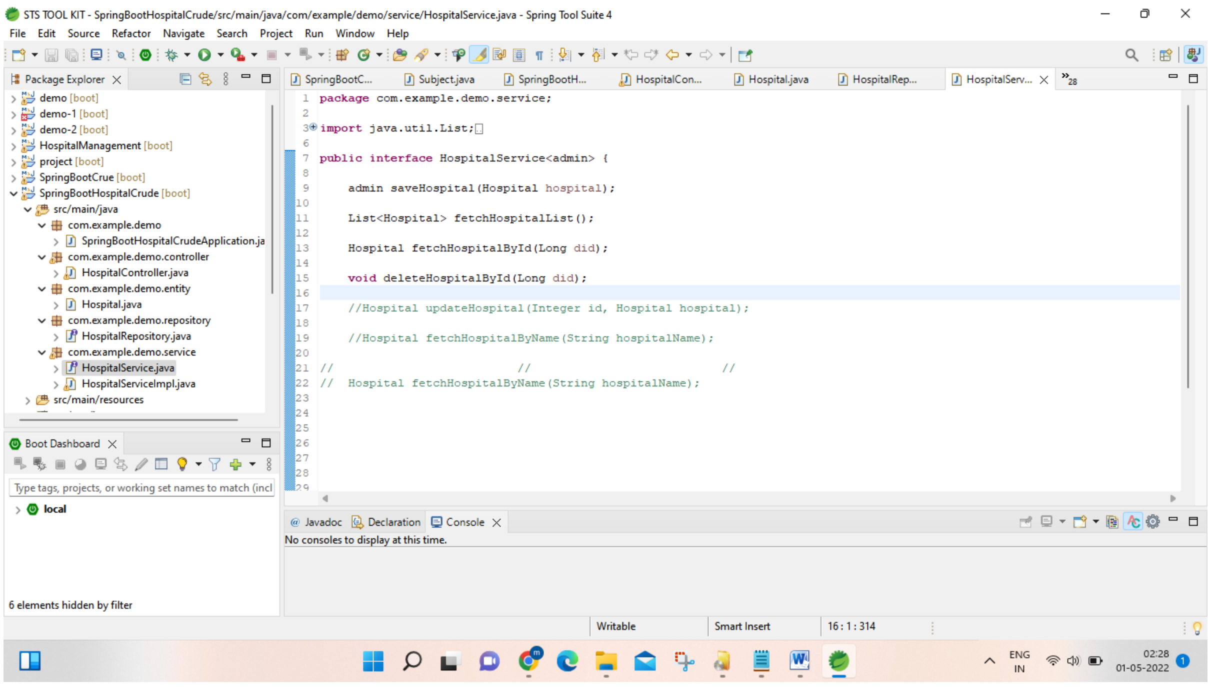


STS TOOL KIT - SpringBootHospitalCrude/src/main/java/com/example/demo/repository/HospitalRepository.java - Spring Tool Suite 4

```java
package com.example.demo.repository;
import org.springframework.data.jpa.repository.JpaRepository;
public interface HospitalRepository extends JpaRepository<Hospital, Long> {



    //Hospital findByHospitalName(String hospitalName);


}
```

No consoles to display at this time.

service with interface name HospitalService

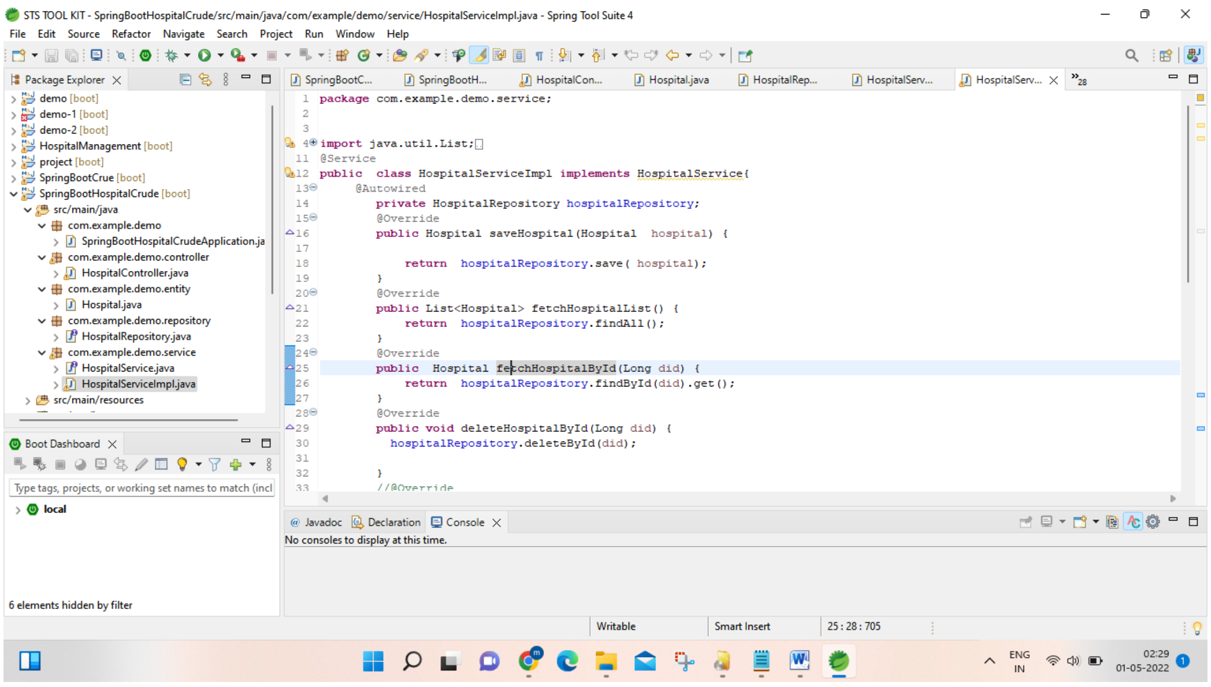


STS TOOL KIT - SpringBootHospitalCrude/src/main/java/com/example/demo/service/HospitalService.java - Spring Tool Suite 4

```java
package com.example.demo.service;

import java.util.List;

public interface HospitalService<admin> {

    admin saveHospital(Hospital hospital);

    List<Hospital> fetchHospitalList();

    Hospital fetchHospitalById(Long did);

    void deleteHospitalById(Long did);

    //Hospital updateHospital(Integer id, Hospital hospital);

    //Hospital fetchHospitalByName(String hospitalName);

//                          //                          //
//  Hospital fetchHospitalByName(String hospitalName);
```

No consoles to display at this time.

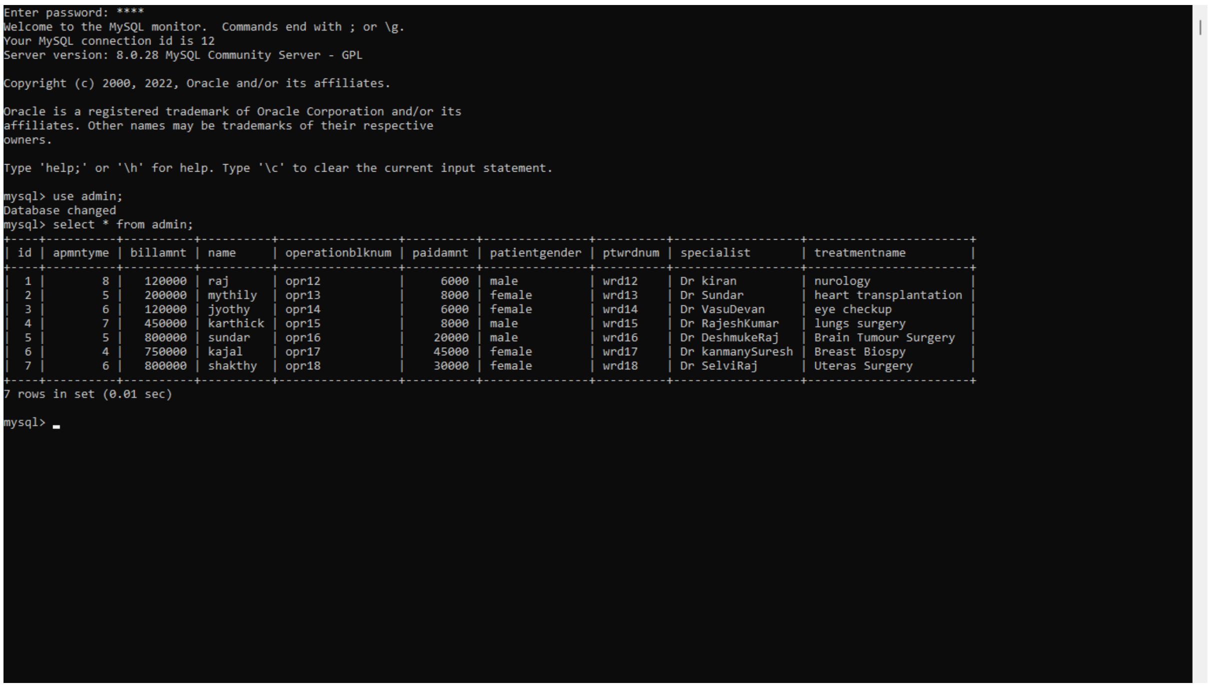
service with class name HospitalServiceImpl



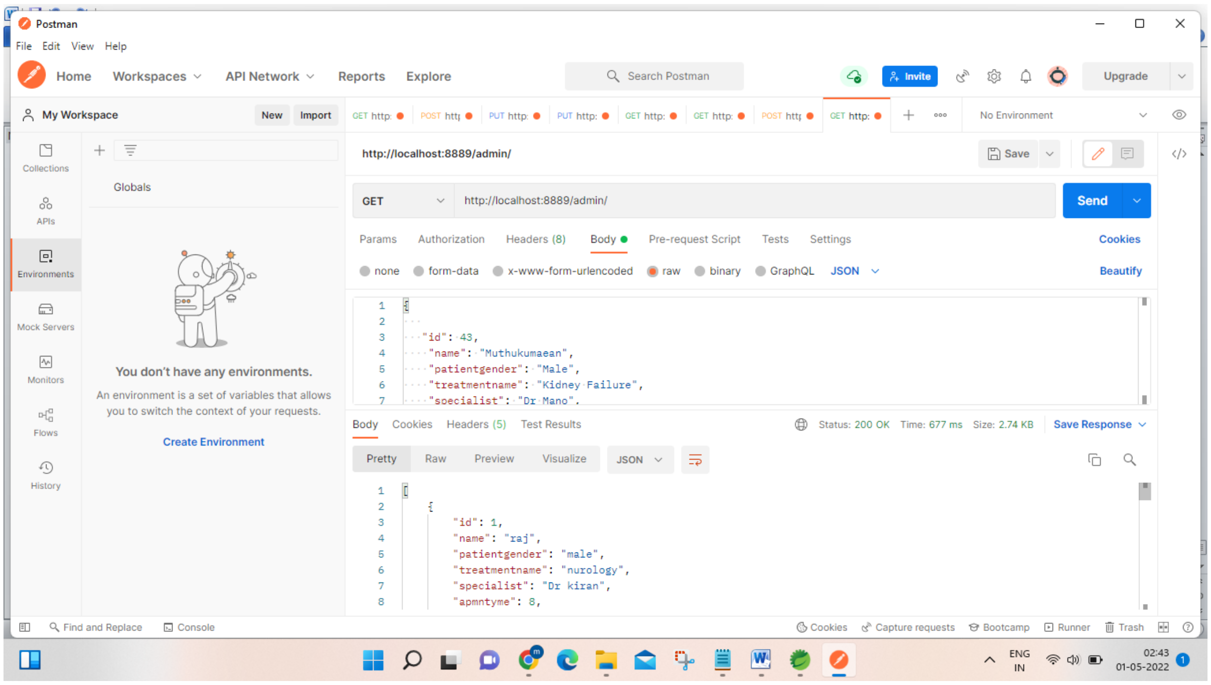
```
package com.example.demo.service;


import java.util.List;
@Service
public  class HospitalServiceImpl implements HospitalService{
     @Autowired
        private HospitalRepository hospitalRepository;
        @Override
        public Hospital saveHospital(Hospital  hospital) {

            return  hospitalRepository.save( hospital);
        }
        @Override
        public List<Hospital> fetchHospitalList() {
            return  hospitalRepository.findAll();
        }
        @Override
        public  Hospital  fetchHospitalById(Long did) {
            return  hospitalRepository.findById(did).get();
        }
        @Override
        public void deleteHospitalById(Long did) {
          hospitalRepository.deleteById(did);

        }
        //@Override
```

```
No consoles to display at this time.
```

FRONT END AND BACK END SUUPORTER



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use admin;
Database changed
mysql> select * from admin;
+----+----------+----------+----------+-----------------+----------+---------------+----------+-----------------+------------------------+
| id | apmntyme | billamnt | name     | operationblknum | paidamnt | patientgender | ptwrdnum | specialist      | treatmentname          |
+----+----------+----------+----------+-----------------+----------+---------------+----------+-----------------+------------------------+
|  1 |        8 |   120000 | raj      | opr12           |     6000 | male          | wrd12    | Dr kiran        | nurology               |
|  2 |        5 |   200000 | mythily  | opr13           |     8000 | female        | wrd13    | Dr Sundar       | heart transplantation  |
|  3 |        6 |   120000 | jyothy   | opr14           |     6000 | female        | wrd14    | Dr VasuDevan    | eye checkup            |
|  4 |        7 |   450000 | karthick | opr15           |     8000 | male          | wrd15    | Dr RajeshKumar  | lungs surgery          |
|  5 |        5 |   800000 | sundar   | opr16           |    20000 | male          | wrd16    | Dr DeshmukeRaj  | Brain Tumour Surgery   |
|  6 |        4 |   750000 | kajal    | opr17           |    45000 | female        | wrd17    | Dr kanmanySuresh| Breast Biospy          |
|  7 |        6 |   800000 | shakthy  | opr18           |    30000 | female        | wrd18    | Dr SelviRaj     | Uteras Surgery         |
+----+----------+----------+----------+-----------------+----------+---------------+----------+-----------------+------------------------+
7 rows in set (0.01 sec)

mysql> _
```


GET http://localhost:8889/admin/

```
{
   
   "id": 43,
    "name": "Muthukumaean",
    "patientgender": "Male",
    "treatmentname": "Kidney Failure",
    "specialist": "Dr Mano",
```

Status: 200 OK Time: 677 ms Size: 2.74 KB

```
[
    {
        "id": 1,
        "name": "raj",
        "patientgender": "male",
        "treatmentname": "nurology",
        "specialist": "Dr kiran",
        "apmntyme": 8,
```

Conclusion:

Registeration of patient details and patient treatment details using sts code run at application properties ,and show the details about patient using database saver in sql ,and enter the informations about patients details using postman app ,not only enter the details and also we can perform various operations like update the patient details,delete the patient details if the patient had discharged means .we can able to delete all of those history about patients.we can save and recover all those patients datas with hospital servicing management services in postman app.