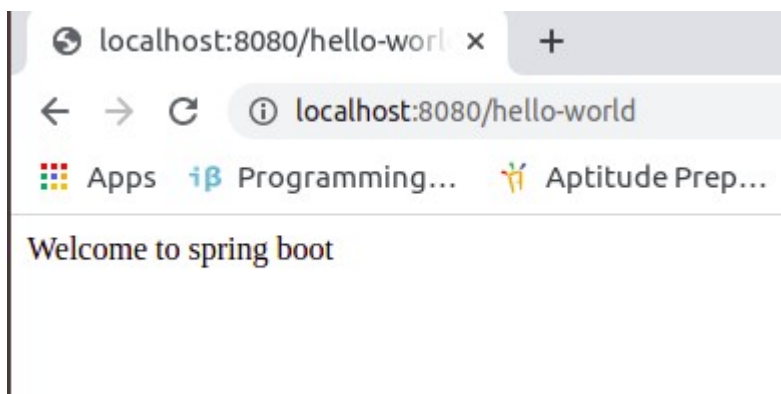
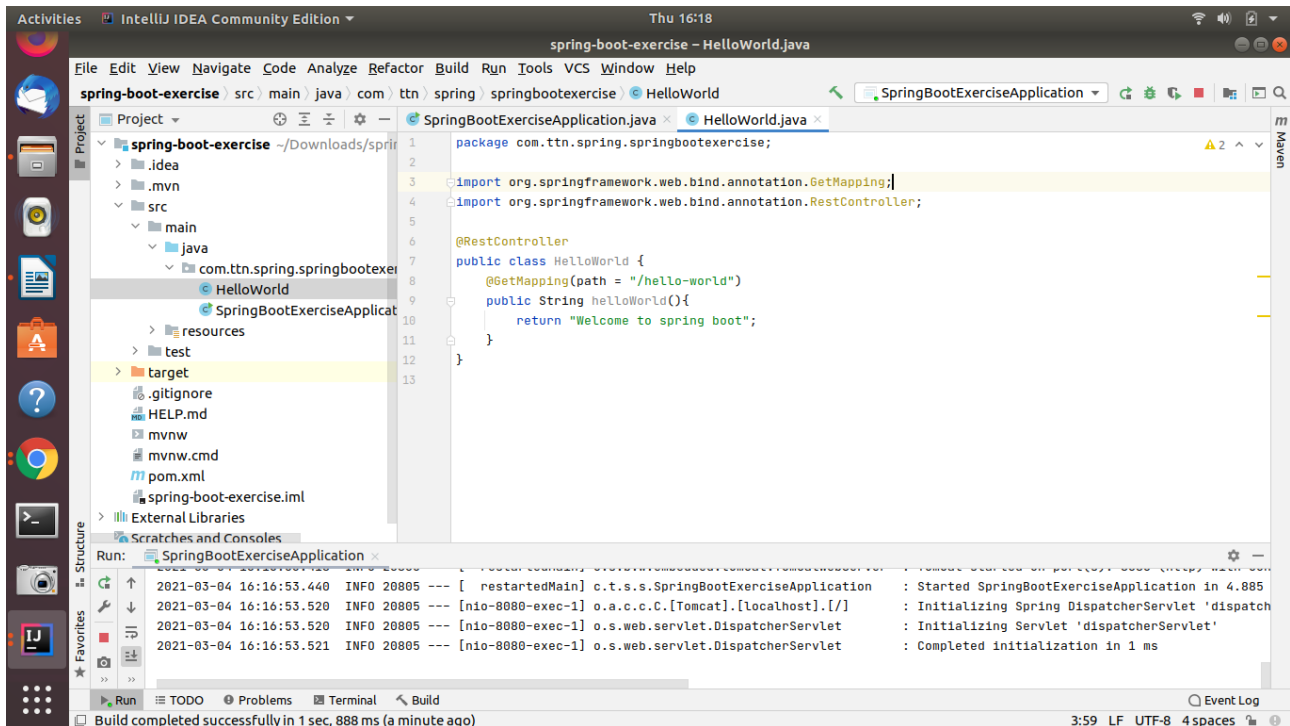


# RestFul Web Service Part 1 Exercise

**Ques1. Create a simple REST ful service in Spring Boot which returns the Response "Welcome to spring boot".**



**Ques2. Create an Employee Bean(id, name, age) and service to perform different operations related to employee.**

```
package com.ttn.spring.springbootexercise.employee;
```

```
public class Employee {
    private Integer id;
    private String name;
    private Integer age;
    public Employee(Integer id, String name, Integer age) {
        this.id = id;
    }
}
```

```

        this.name = name;
        this.age = age;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Employee{" +
            "id=" + id +
            ", name=" + name + " +
            ", age=" + age +
            '}';
    }
}

```

```

package com.ttn.spring.springbootexercise.employee;

import org.springframework.stereotype.Component;
import java.util.ArrayList;
import java.util.List;
@Component
public class EmployeeDaoService {
    private static List<Employee> employees = new ArrayList<>();
    private static int employeeCount = 3;
    static {
        employees.add(new Employee(1, "Naveen", 24));
        employees.add(new Employee(2, "Manoj", 26));
        employees.add(new Employee(3, "Kajal", 23));
    }
    public List<Employee> findAll() {
        return employees;
    }
    public Employee save(Employee emp) {
        if (emp.getId() == null) {
            emp.setId(++employeeCount);
        }
        employees.add(emp);
        return emp;
    }
    public Employee findOne(int id) {
        for (Employee emp : employees) {
            if (emp.getId() == id) {

```

```

        return emp;
    }
}
return null;
}
}

```

```

package com.ttn.spring.springbootexercise.employee;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;

@RestController
public class EmployeeResource {
    @Autowired
    private EmployeeDaoService service;
    @GetMapping("/employees")
    public List<Employee> retrieveAllUser(){
        return service.findAll();
    }
}

```

**Ques. 3 Implement GET http request for Employee to get list of employees.**

```

package com.ttn.spring.springbootexercise.employee;

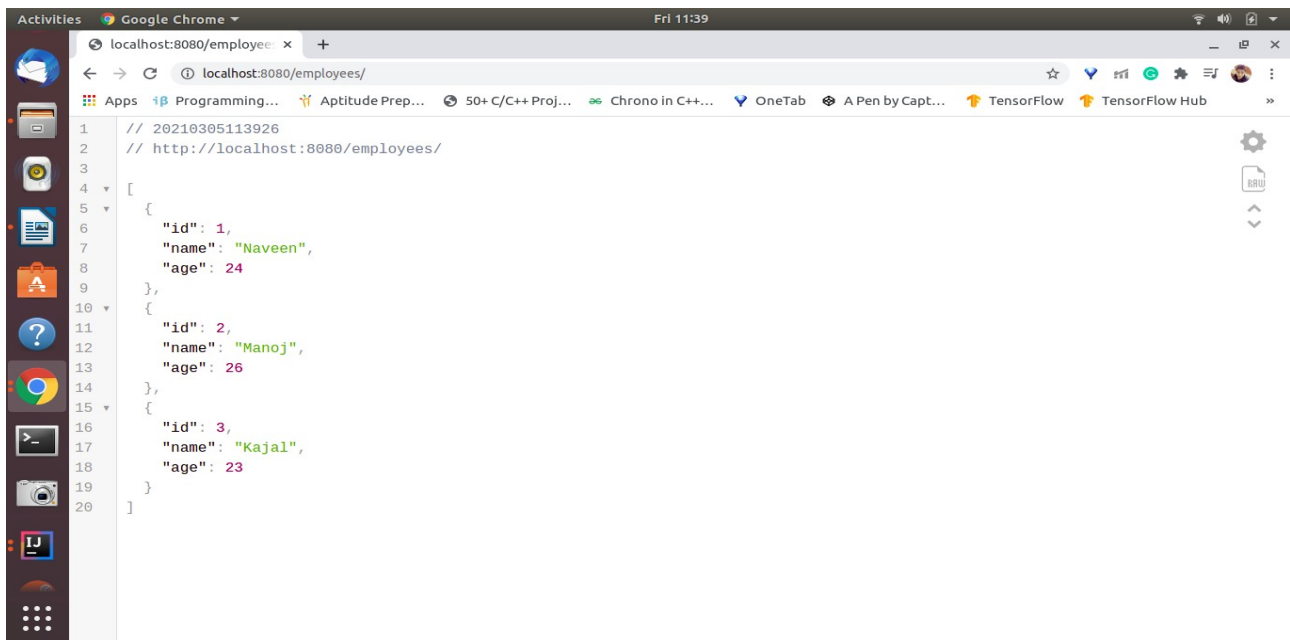
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class EmployeeResource {
    @Autowired
    private EmployeeDaoService service;

    @GetMapping("/employees")
    public List<Employee> retrieveAllUser(){
        return service.findAll();
    }
}

```



#### Ques 4. Implement GEThttp request using path variable top get one employee

```
package com.ttn.spring.springbootexercise.employee;

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

import java.util.List;

@RestController
public class EmployeeResource {
    @Autowired
    private EmployeeDaoService service;

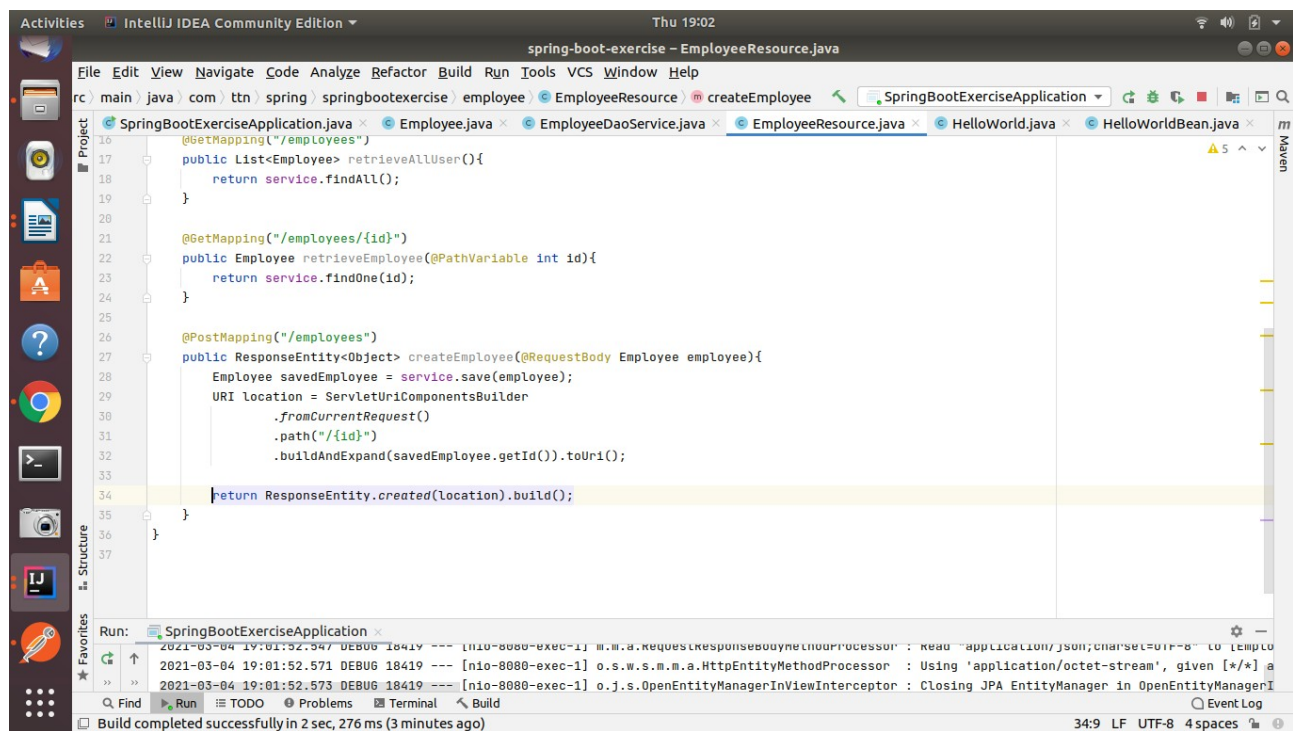
    @GetMapping("/employees")
    public List<Employee> retrieveAllUser(){
        return service.findAll();
    }

    @GetMapping("/employees/{id}")
    public Employee retrieveEmployee(@PathVariable int id){
        return service.findOne(id);
    }
}
```

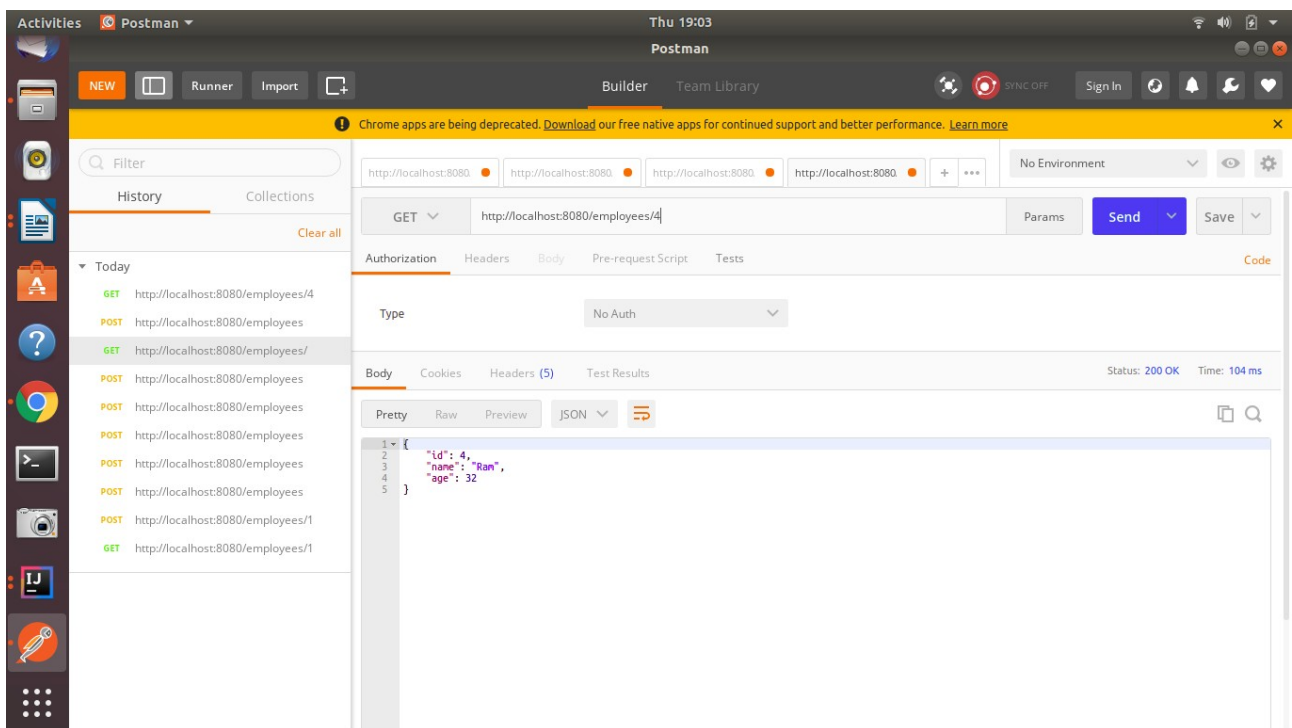
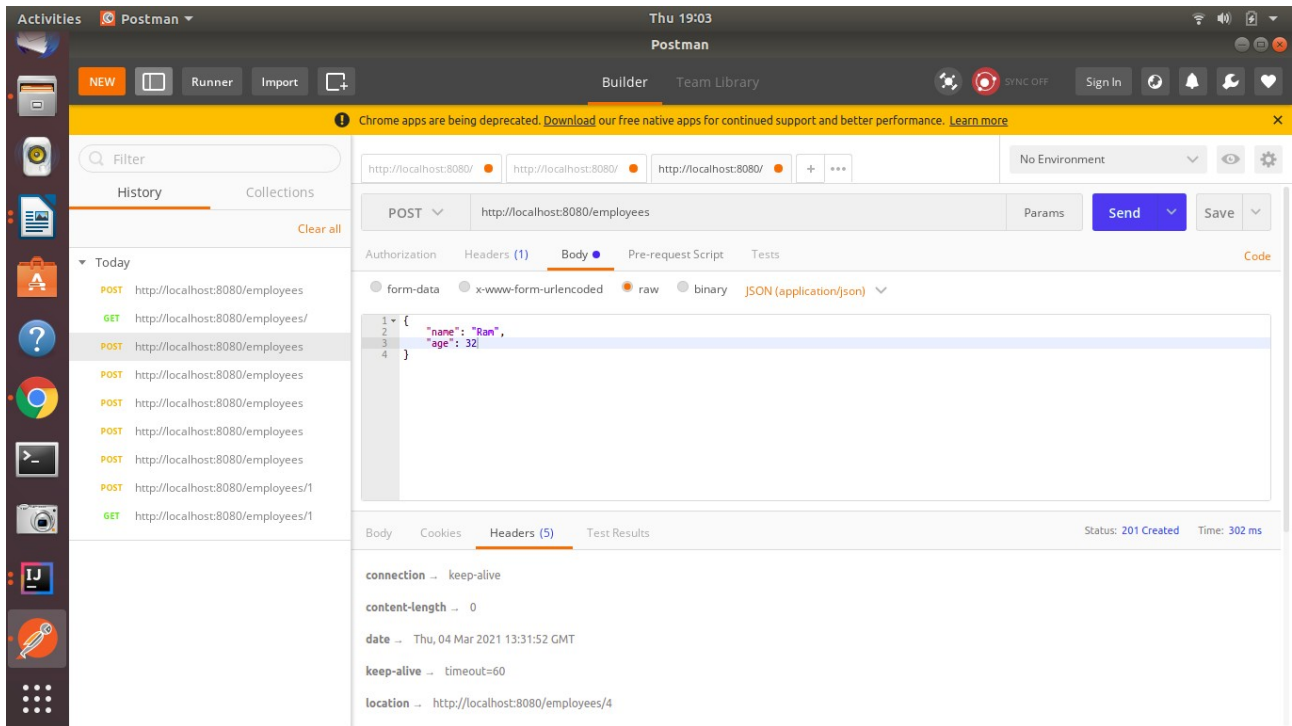


```
localhost:8080/employees x +
localhost:8080/employees/1
Apps Programming... Aptitude Prep... 50+ C/C++ Proj
1 // 20210305114022
2 // http://localhost:8080/employees/1
3
4 {
5     "id": 1,
6     "name": "Naveen",
7     "age": 24
8 }
```

**Ques5. Implement POST http request for Employee to create a new employee.**



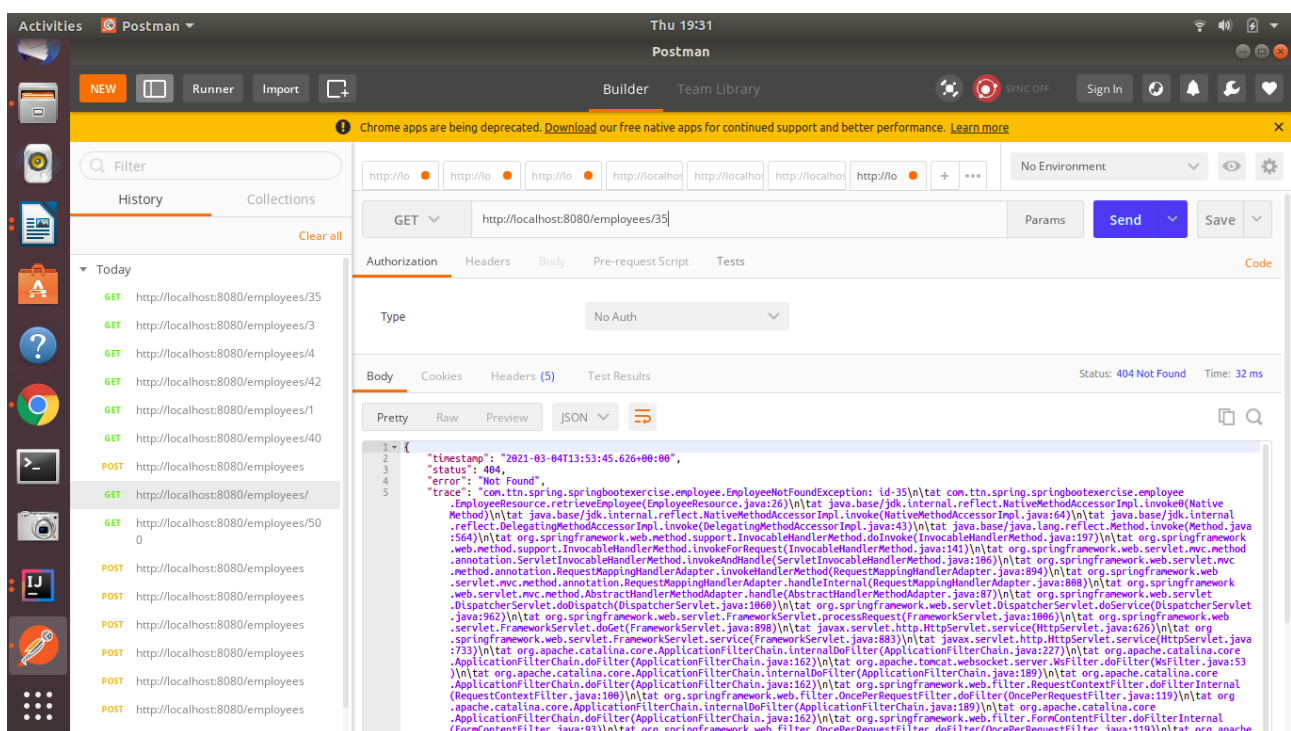
```
IntelliJ IDEA Community Edition Thu 19:02
spring-boot-exercise - EmployeeResource.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
rc main java com ttn spring springbootexercise employee EmployeeResource createEmployee SpringBootExerciseApplication
SpringBootExerciseApplication.java Employee.java EmployeeDaoService.java EmployeeResource.java HelloWorld.java HelloWorldBean.java
16 @GetMapping("/employees")
17 public List<Employee> retrieveAllUser(){
18     return service.findAll();
19 }
20
21 @GetMapping("/employees/{id}")
22 public Employee retrieveEmployee(@PathVariable int id){
23     return service.findOne(id);
24 }
25
26 @PostMapping("/employees")
27 public ResponseEntity<Object> createEmployee(@RequestBody Employee employee){
28     Employee savedEmployee = service.save(employee);
29     URI location = ServletUriComponentsBuilder
30         .fromCurrentRequest()
31         .path("/{id}")
32         .buildAndExpand(savedEmployee.getId()).toUri();
33
34     return ResponseEntity.created(location).build();
35 }
36
37
Run: SpringBootExerciseApplication
2021-03-04 19:01:52.547 DEB 18419 --- [nio-8080-exec-1] m.m.a.RequestResponseBodyMethodProcessor : read [application/json;charset=utf-8] to [empto
2021-03-04 19:01:52.571 DEB 18419 --- [nio-8080-exec-1] o.s.w.s.m.a.HttpEntityMethodProcessor : Using 'application/octet-stream', given [*/] a
2021-03-04 19:01:52.573 DEB 18419 --- [nio-8080-exec-1] o.j.s.OpenEntityManagerInViewInterceptor : Closing JPA EntityManager in OpenEntityManager
Build completed successfully in 2 sec, 276 ms (3 minutes ago) 34:9 LF UTF-8 4 spaces
```





## Ques 6. Implement Exception Handling for resource not found

```
@GetMapping("/employees/{id}")
public Employee retrieveEmployee(@PathVariable int id){
    Employee emp = service.findOne(id);
    if (emp == null)
        throw new EmployeeNotFoundException("id-" + id);
    return emp;
}
```



## Ques 7. Implement DELETE http request for Employee to delete employee

```
public Employee deleteById(int id) {
    Iterator<Employee> iterator = employees.iterator();
    while(iterator.hasNext()) {
        Employee emp = iterator.next();
        if (emp.getId() == id) {
            iterator.remove();
            return emp;
        }
    }
    return null;
}
```

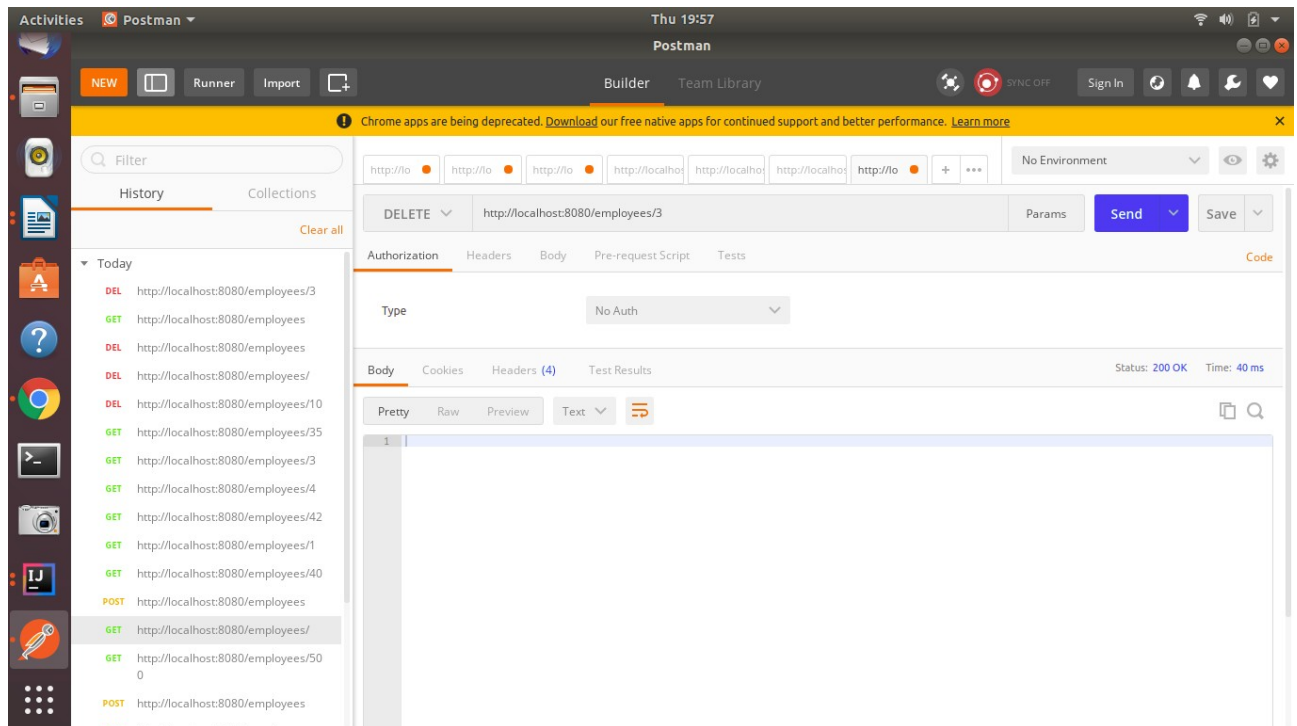
```

@DeleteMapping("/employees/{id}")
public void deleteEmployee(@PathVariable int id){
    Employee emp = service.deleteById(id);
    if (emp == null)
        throw new EmployeeNotFoundException("id-" + id);
}
}
}

```

The screenshot shows a Linux desktop with a sidebar containing icons for applications like a file manager, terminal, and web browser. The main window is Postman, a REST client. The interface includes a top bar with 'NEW', 'Runner', 'Import', and 'Builder' buttons. Below this is a search bar and a 'Filter' button. The left sidebar shows a 'History' tab with a list of recent requests, including GET and DELETE requests to 'http://localhost:8080/employees/'. The main area displays a GET request to 'http://localhost:8080/employees/'. The 'Body' tab is selected, showing a JSON array of two employee objects: one with id 1, name 'Naveen', and age 24; and another with id 2, name 'Manoj', and age 26. The status bar at the bottom of the Postman window indicates 'Status: 200 OK' and 'Time: 21 ms'.



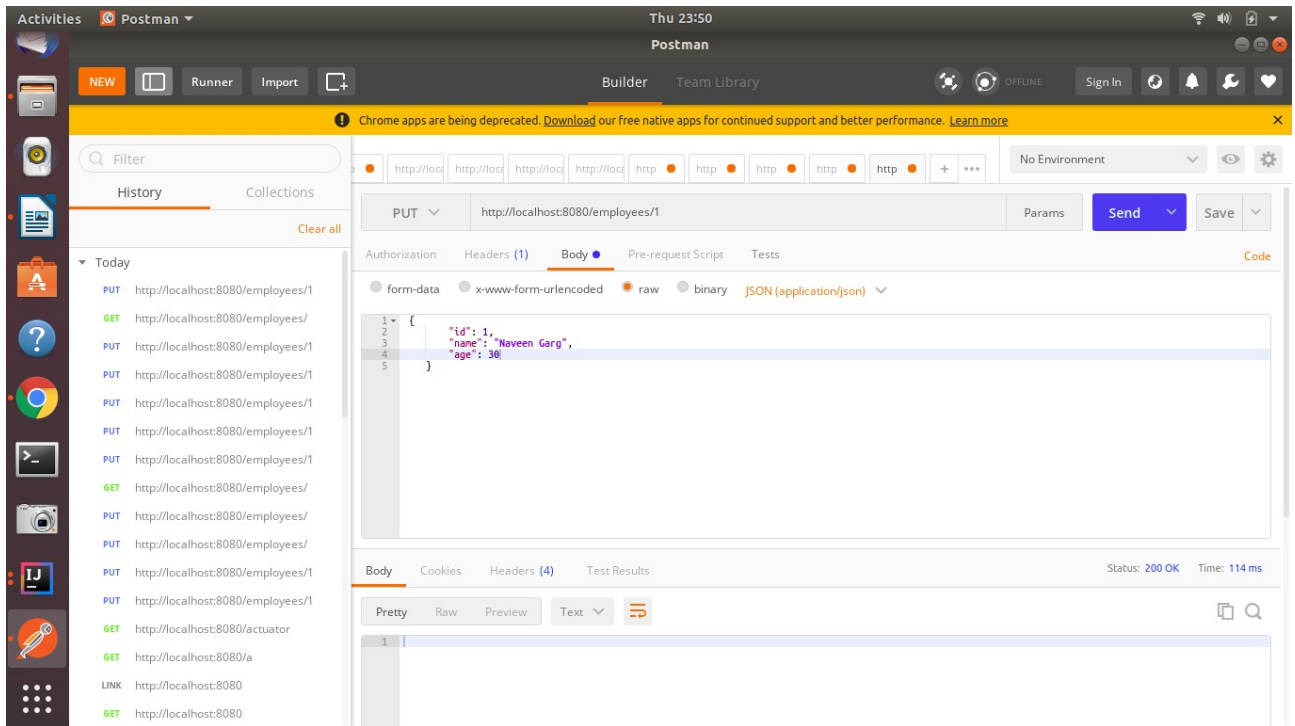
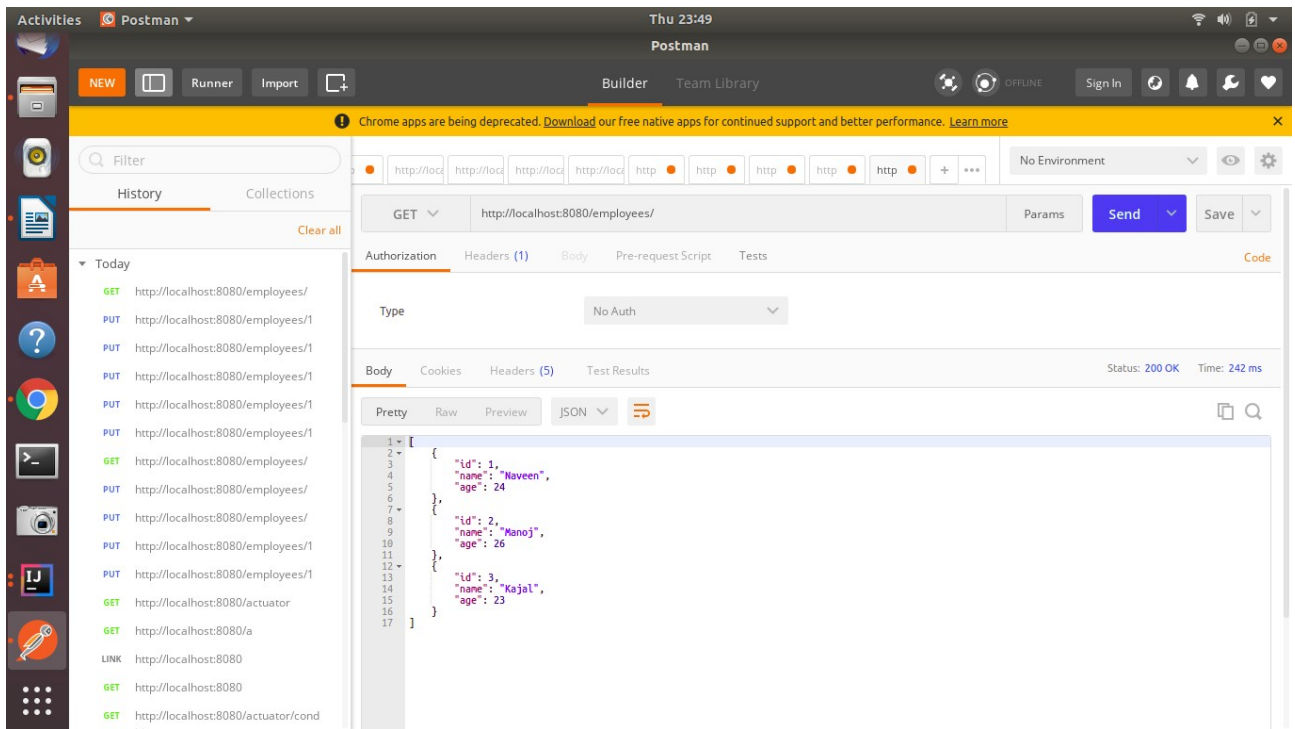


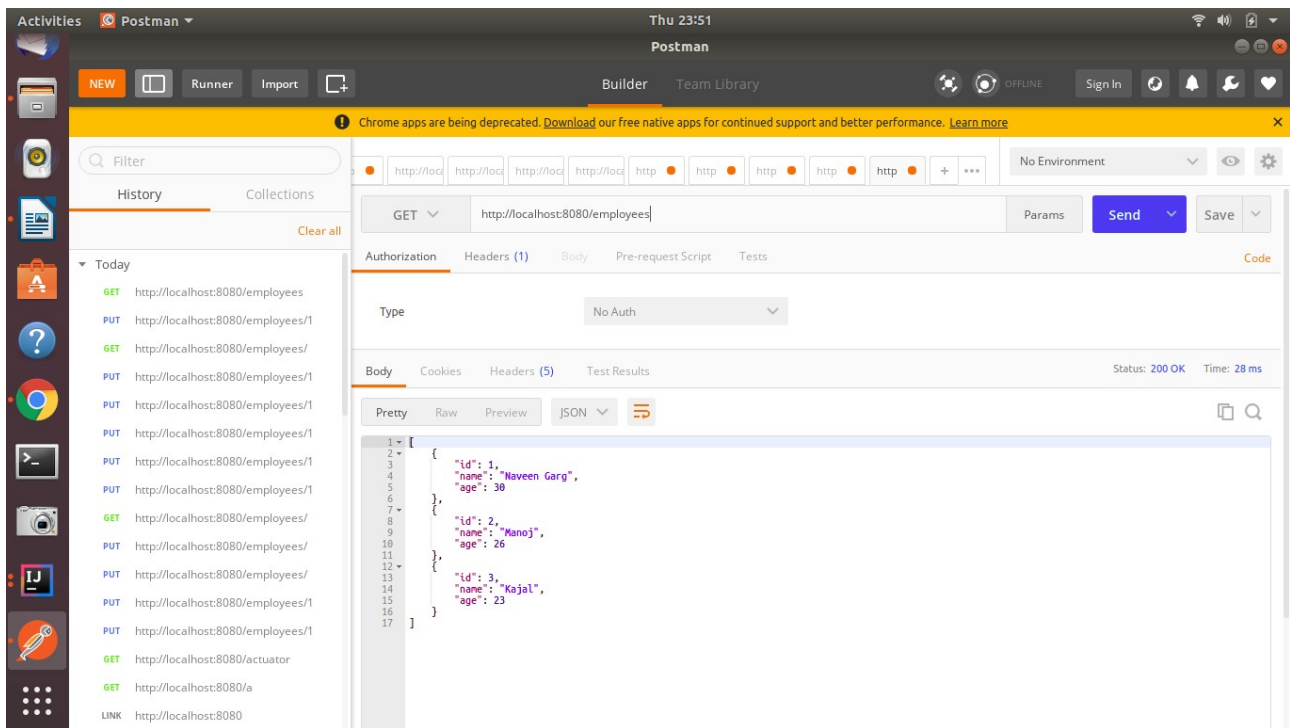
### Ques 8. Implement PUT http request for Employee to update employee

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

```
@PutMapping("/employees/{id}")
public void updateEmployee(@RequestBody Employee emp, @PathVariable int id){
    service.updateEmployee(id, emp);
}
```

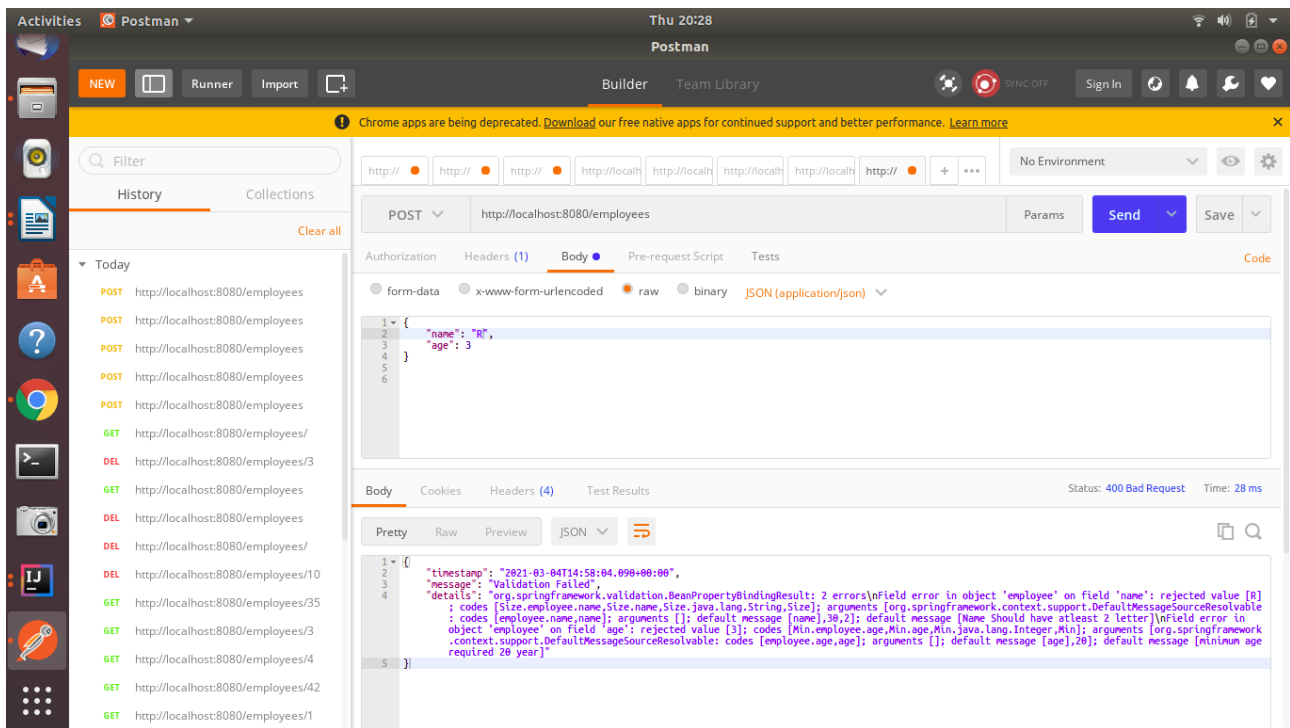
```
public void updateEmployee(Integer id, Employee emp) {
    for(int i=0; i<employees.size(); i++){
        Employee e = employees.get(i);
        if(e.getId().equals(id)){
            employees.set(i, emp);
            return;
        }
    }
}
```





**Ques 9. Apply validation while create a new employee using POST http Request.**

```
1 package com.ttn.spring.springbootexercise.employee;  
2  
3 import org.springframework.beans.factory.annotation.Value;  
4  
5 import javax.validation.constraints.Min;  
6 import javax.validation.constraints.NotNull;  
7 import javax.validation.constraints.Size;  
8  
9 public class Employee {  
10     private Integer id;  
11     @NotNull  
12     @Size(min=2, max=30, message = "Name Should have atleast 2 letter")  
13     private String name;  
14  
15     @NotNull  
16     @Min(value = 20, message = "minimum age required 20 |year")  
17     private Integer age;  
18  
19     public Employee() {  
20  
21     }  
22 }
```



**Ques 10. Configure actuator in your project to check the health of application and get the information about various beans configured in your application**

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-data-rest-hal-browser -->
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-rest-hal-browser</artifactId>
  <version>3.3.7.RELEASE</version>
</dependency>
```

Activities Google Chrome Fri 11:42

localhost:8080/actuator

```
1 // 20210305114247
2 // http://localhost:8080/actuator
3
4 {
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/actuator",
8       "templated": false
9     },
10    "beans": {
11      "href": "http://localhost:8080/actuator/beans",
12      "templated": false
13    },
14    "caches-cache": {
15      "href": "http://localhost:8080/actuator/caches/{cache}",
16      "templated": true
17    },
18    "caches": {
19      "href": "http://localhost:8080/actuator/caches",
20      "templated": false
21    },
22    "health": {
23      "href": "http://localhost:8080/actuator/health",
24      "templated": false
25    },
26    "health-path": {
27      "href": "http://localhost:8080/actuator/health/{path}"
```

localhost:8080/actuator localhost:8080/actuator/health

```
1 // 20210305114312
2 // http://localhost:8080/actuator/health
3
4 {
5   "status": "UP"
6 }
```

Activities Google Chrome Fri 11:44

localhost:8080/actuator localhost:8080/actuator/beans

```
1 // 20210305114339
2 // http://localhost:8080/actuator/beans
3
4 {
5   "contexts": {
6     "application": {
7       "beans": {
8         "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
9           "aliases": [
10             ],
11           "scope": "singleton",
12           "type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
13           "resource": null,
14           "dependencies": [
15             ],
16           ],
17       },
18     },
19     "pagedResourcesAssembler": {
20       "aliases": [
21         ],
22       },
23     },
24     "applicationTaskExecutor": {
25       "aliases": [
26         "taskExecutor"
27       ],
28       "scope": "singleton",
29       "type": "org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor",
30       "resource": "class path resource [org.springframework.boot.autoconfigure.task/TaskExecutionAutoConfiguration.class]",
31       "dependencies": [
32         "org.springframework.boot.autoconfigure.task.TaskExecutionAutoConfiguration",
33         "taskExecutorBuilder"
34       ],
35     },
36   ],
37 }
```