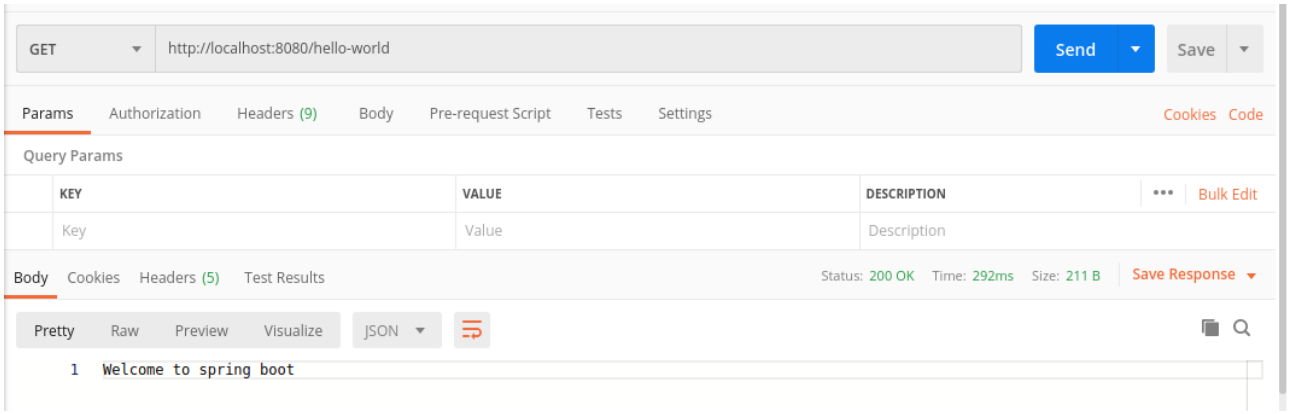
RestFulWeb Service Part-1

Q1Create a simple REST fulservice in Spring Boot which returns the Response "Welcome to spring boot".

```
@RestController
public class HelloWorldController {
    @GetMapping(path = "/hello-world")
    public String helloWorld(){
        return "Welcome to spring boot";
    }
}
```



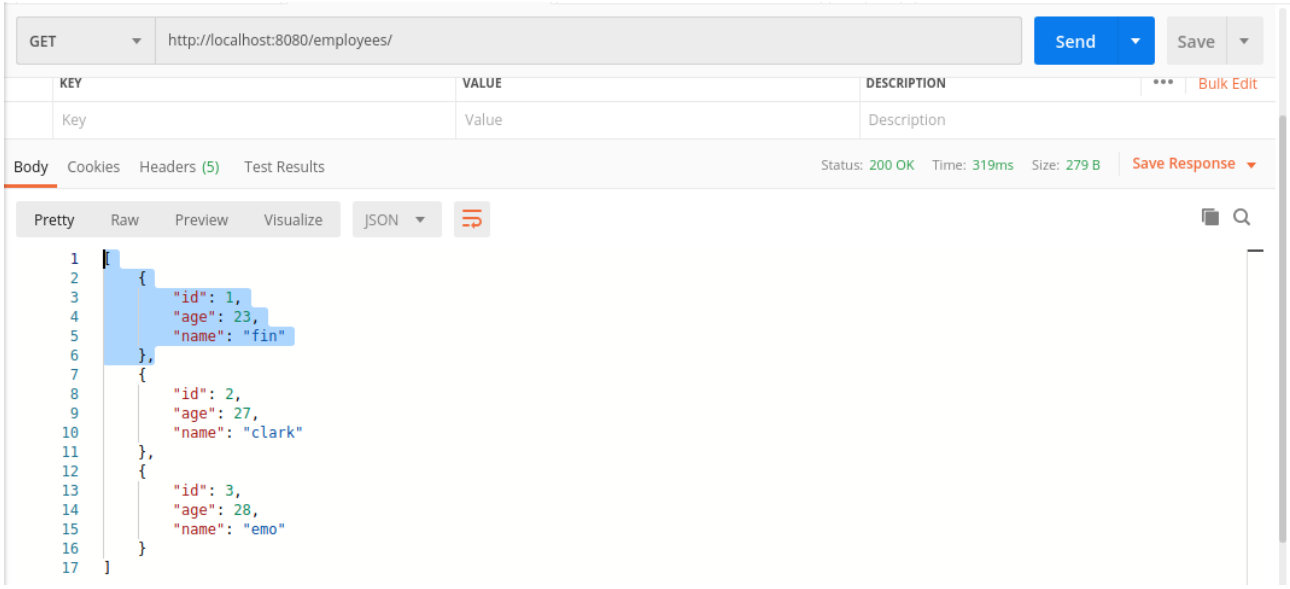
Q2Create an Employee Bean(id, name, age) and service to perform different operations related to employee.

```
package com.bootcampassignment.RestfulWebService.Q2.entity;
import org.hibernate.validator.constraints.Range;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
public class Employee {
    private Integer id;
    @Range(min = 18, message = "minimum age should be 18")
    private int age;
    @Size(min = 2, message = "name should be greater than 2
characters")
    private String name;
    Employee() {
    }
    public Employee(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Q3Implement GET http request for Employee to get list of employees.

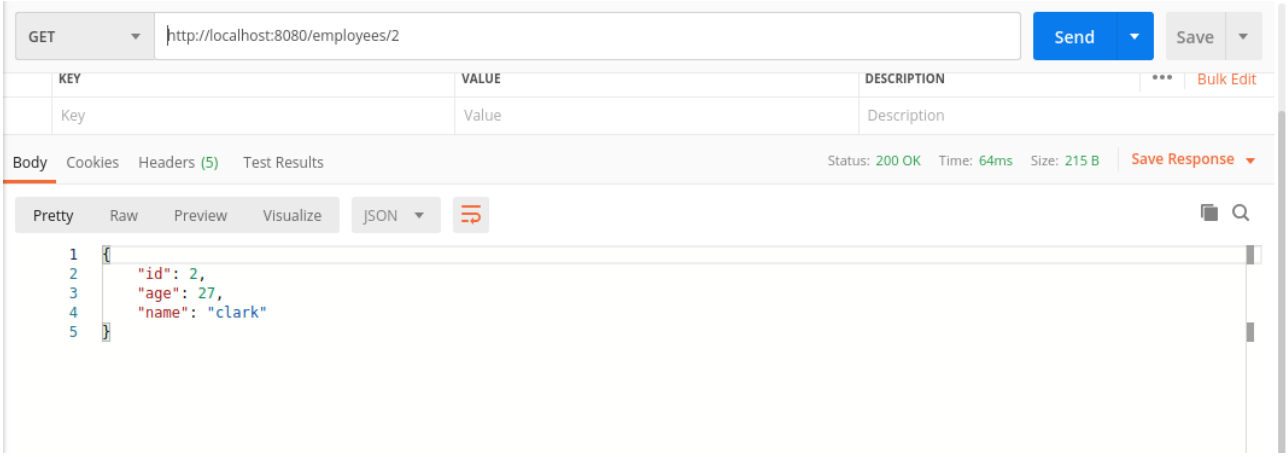
```
@RestController
@RequestMapping(path = "/employees")
public class EmployeeController {
    @Autowired
    private EmployeeService empquerry;
    // to see list of all employee
    @GetMapping(path = "/")
    public ResponseEntity<List<Employee>> requestEmpList() {
        return new ResponseEntity(empquerry.getAllEmp(),
HttpStatus.OK);
    }

@Component
public class EmployeeService {
    private static List<Employee> empList = new ArrayList<>();
    static int empCount = 3;
    static {
        empList.add(new Employee(1, "fin", 23));
        empList.add(new Employee(2, "clark", 27));
        empList.add(new Employee(3, "emo", 28));
    }
    // get list of all employees
    public List<Employee> getAllEmp() {
        return empList;
    }
```



Q4Implement GET http request using path variable top get one employee.

```
// to see a specific employee
@GetMapping(path = "/{id}")
public Employee requestEmp(@PathVariable Integer id) {
    Employee foundedEmp=empquerry.getEmp(id);
    if(foundedEmp==null){
        throw new UserNotFoundException("User Not Found With id-
>"+id);
    }
    return foundedEmp;
}
```

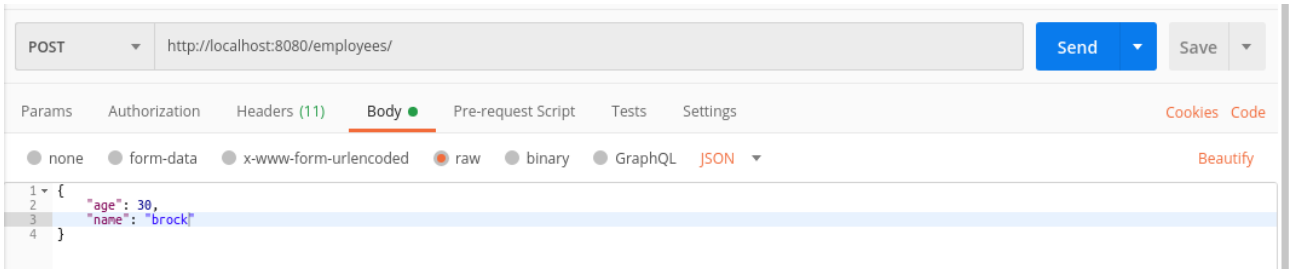
```
    // get a specific employee from the list
    public Employee getEmp(Integer id) {
        for (Employee emp : empList) {
            if (emp.getId() == id) {
                return emp;
            }
        }
        return null;
    }
```

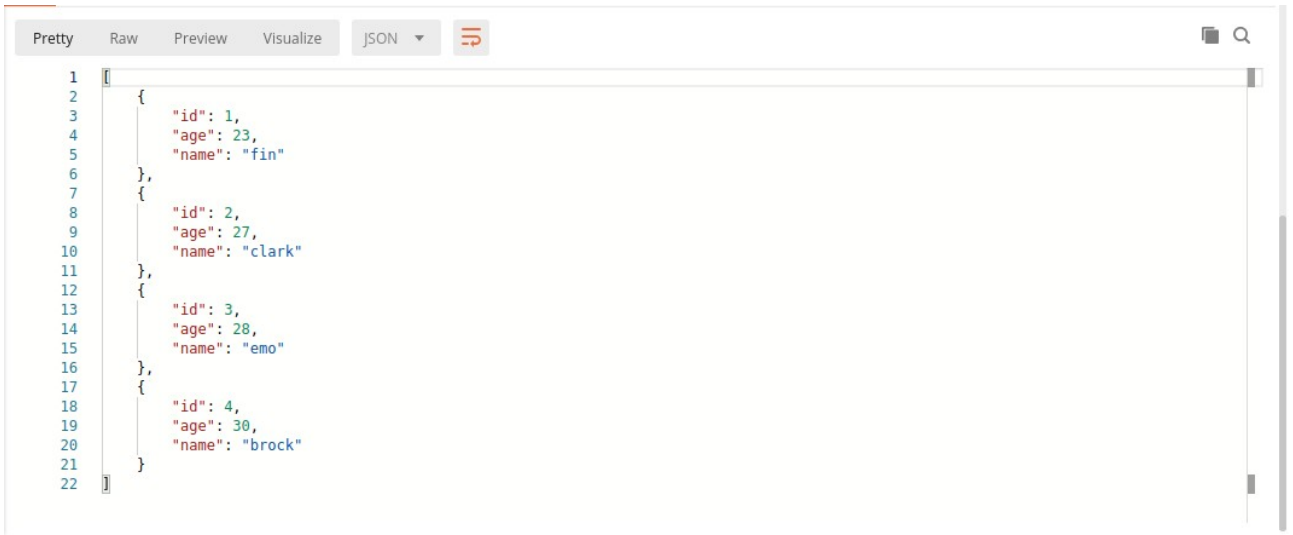


Q5Implement POST http request for Employee to create a new employee.

```
@PostMapping(path = "/")
public ResponseEntity<Object> createEmp(@Valid @RequestBody
Employee emp) {
    Employee savedEmployee = empquerry.addEmp(emp);
    //returning the status code
    URI
location=ServletUriComponentsBuilder.fromCurrentRequestUri()//
return /users
            .path("/{id}") //add path to uri->/users/{id}
            .buildAndExpand(savedEmployee.getId()) // replace d with
{id}-> employee id
            .toUri();// convert it to uri
    return ResponseEntity.created(location).build(); // returning
the created status
}
```

```
  // add employee to the list
    public Employee addEmp(Employee emp) {
        if (emp.getId() == null) {
            emp.setId(empCount += 1);
        }

            empList.add(emp);
            return emp;
    }
```

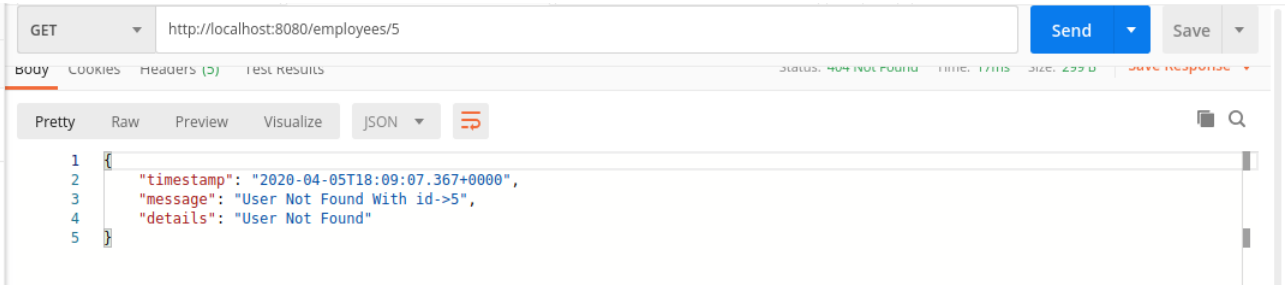




Q6Imploement Exception Handling for resource not found

```
package com.bootcampassignment.RestfulWebService.Q2.exception;
import java.util.Date;
public class ExceptionResponse {
    private Date timestamp;
    private String message;
    private String details;
    ExceptionResponse(Date timestamp,String message,String details)
{
        this.timestamp=timestamp;
        this.message=message;
        this.details=details;
    }
    public Date getTimestamp(){
        return timestamp;
    }
    public String getMessage(){
        return message;
    }
    public  String getDetails(){
        return details;
    }
}

@ResponseStatus(HttpStatus.NOT_FOUND)
```

```
public class UserNotFoundException extends RuntimeException {
    public UserNotFoundException(String s) {
        super(s);
    }
}

package com.bootcampassignment.RestfulWebService.Q2.exception;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import
org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEnti
tyExceptionHandler;
import java.util.Date;
@RestControllerAdvice
public class EmployeeExceptionHandler extends
ResponseEntityExceptionHandler {
    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object>
handleAllException(Exception ex, WebRequest request){
        ExceptionResponse exr;
        exr=new ExceptionResponse(new
Date(),ex.getMessage(),request.getDescription(false));
        return new ResponseEntity(exr,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
    @ExceptionHandler(UserNotFoundException.class)
    public final ResponseEntity<Object>
handleUserNotFoundException(UserNotFoundException ex, WebRequest
request){
        ExceptionResponse exr;
        exr=new ExceptionResponse(new Date(),ex.getMessage(),"User
Not Found");
        return new ResponseEntity(exr, HttpStatus.NOT_FOUND);
    }
    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(
            MethodArgumentNotValidException ex, HttpHeaders headers,
HttpStatus status, WebRequest request) {
        ExceptionResponse exr;
        exr=new ExceptionResponse(new
Date(),ex.getMessage(),ex.getBindingResult().toString());
        return new ResponseEntity(exr, HttpStatus.BAD_REQUEST);
    }
}
```

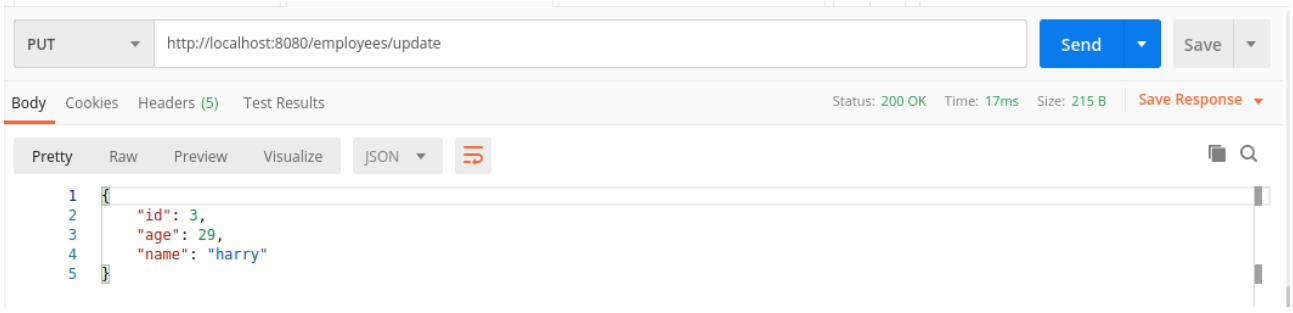


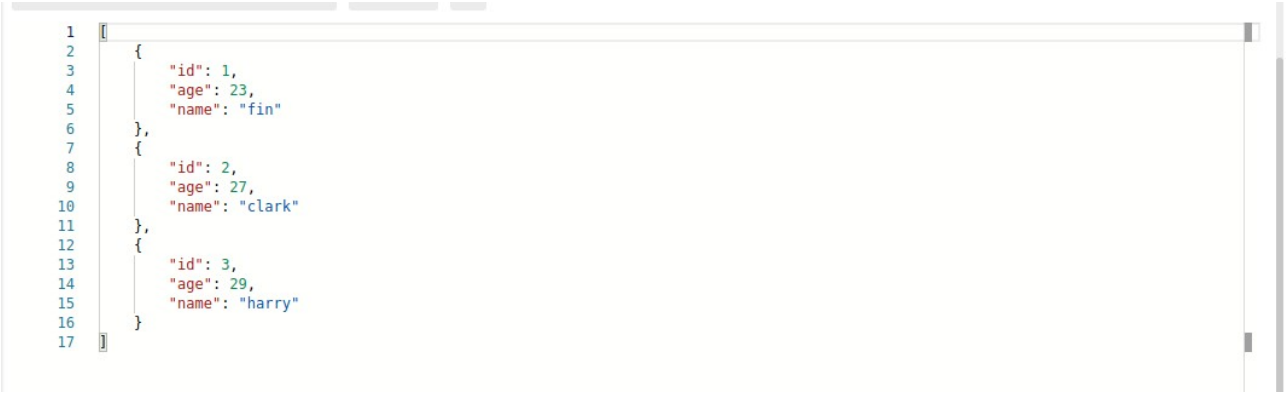
Q7Implement DELETE http request for Employee to delete employee

```
//to delete a employee from the list
@DeleteMapping(path = "/remove/{id}")
public ResponseEntity<String> removeEmployee(@PathVariable Integer
id){
    Employee exemp=empquerry.removeEmp(id);
    if(exemp==null){
        throw new UserNotFoundException("User Not Found With id-
>"+id);
    }
    return new ResponseEntity<>("the record has been
deleted",HttpStatus.OK);
}

// delete a employee record
public Employee removeEmp(Integer id){
    Iterator<Employee> empitr=empList.iterator();
    while (empitr.hasNext()){
        Employee emp=empitr.next();
        if (emp.getId()==id){
            empitr.remove(); // will remove employee from list
            return emp;
        }
    }
    return null;
}
```

Q8Implement PUT http request for Employee to update employee

```
     //to update a employee from the list
    @PutMapping(path = "/update")
    public ResponseEntity<Employee> updateEmployee(@Valid
@RequestBody Employee employee){
        employee = empquerry.promoteEmployee(employee);
        if(employee==null){
            throw new UserNotFoundException("User Not Found With id-
>"+employee.getId());
        }
        return new ResponseEntity<>(employee,HttpStatus.OK);
    }
}

// update existing employee details
public Employee promoteEmployee(Employee employee){
    for (Employee emp:empList){
        if(emp.getId()==employee.getId()){
            emp.setName(employee.getName());
            emp.setAge(employee.getAge());
            return emp;
        }
    }
    return null;
}
```

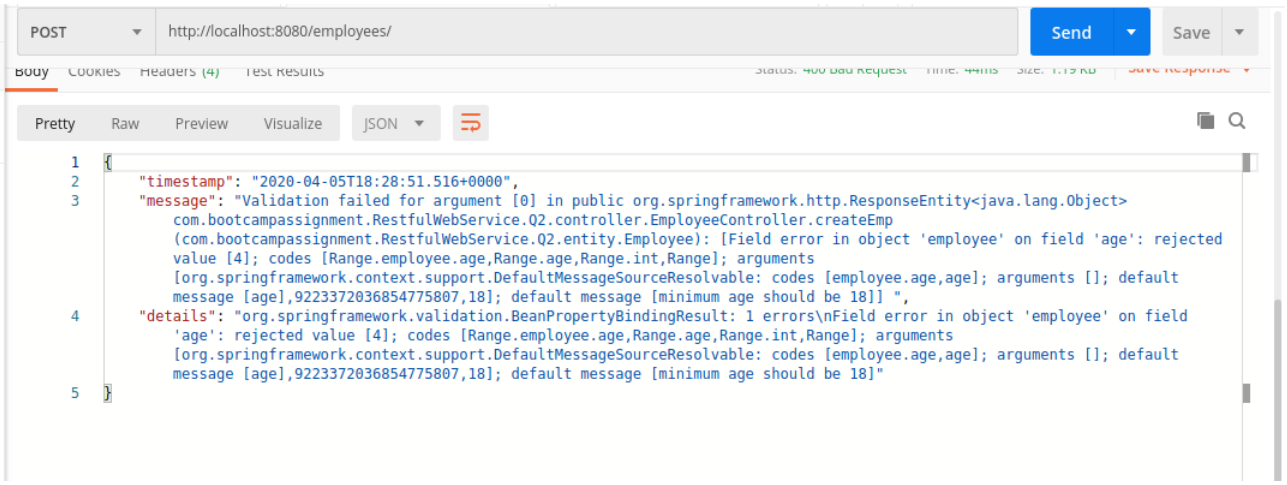




Q9Apply validation while create a new employee using POST http Request.

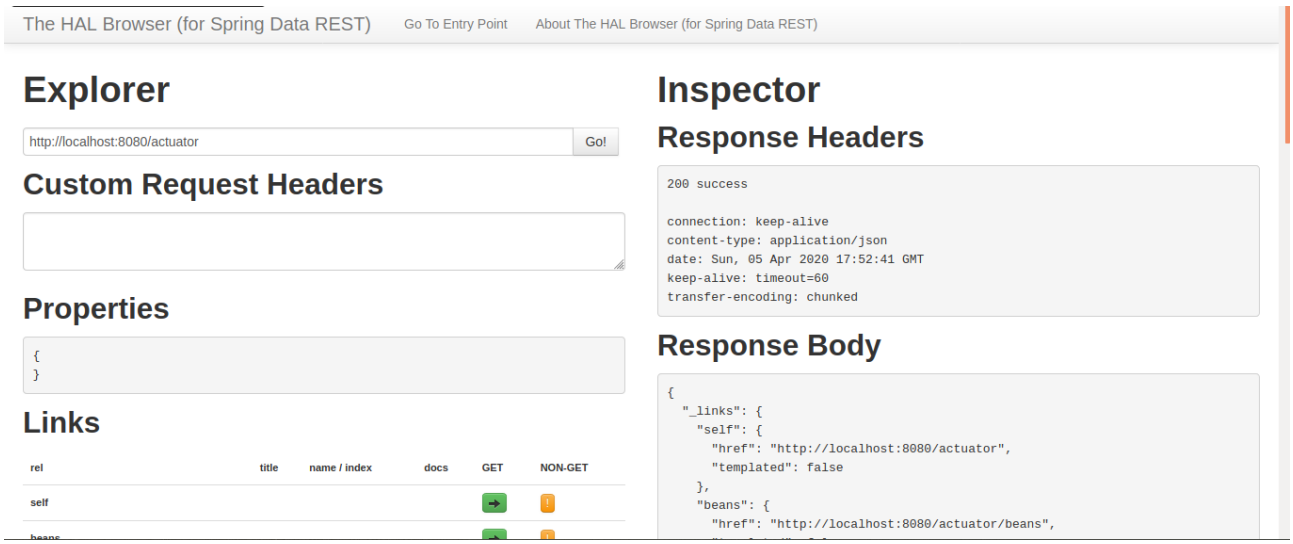
```
public class Employee {
    private Integer id;
    @Range(min = 18, message = "minimum age should be 18")
    private int age;
    @Size(min = 2, message = "name should be greater than 2
characters")
    private String name;
```

// getter and setter



Q10Configure actuator in your project to check the health of application and get the information about various beans configured in your application

```
//custom
compile group: 'org.springframework.data', name: 'spring-data-
rest-hal-browser'

implementation'org.springframework.boot:spring-boot-starter-
actuator'

management.endpoints.web.exposure.include=*
```


The HAL Browser (for Spring Data REST)
Go To Entry Point
About The HAL Browser (for Spring Data REST)
Explorer
http://localhost:8080/actuator
Go!
Custom Request Headers
Properties
{
}
Links
rel
title
name / index
docs
GET
NON-GET
self
Inspector
Response Headers
200 success
connection: keep-alive
content-type: application/json
date: Sun, 05 Apr 2020 17:52:41 GMT
keep-alive: timeout=60
transfer-encoding: chunked
Response Body
{
"_links": {
"self": {
"href": "http://localhost:8080/actuator",
"templated": false
},
"beans": {
"href": "http://localhost:8080/actuator/beans",