Session: RestFul Web Service Part 1

Assignment

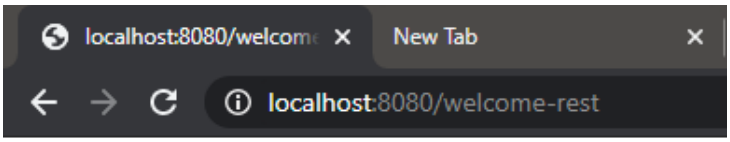
- Create a simple RESTful service in Spring Boot which returns the Response "Welcome to spring boot".

```
@RestController
public class HelloWorldRest {

    //@RequestMapping(method = RequestMethod.GET,path = "/hello-world")
    @GetMapping(path = "/welcome-rest")
    public String getMessageRest() {
        return "Welcome to spring boot";
    }

    //using a seperate bean(HelloWorldBean) for displaying the message
    @GetMapping(path = "/welcome-bean")
    public HelloWorldBean getMessageBean() {
        return new HelloWorldBean("Welcome to spring boot");
    }


    //displaying a name present in the path along with message
    @GetMapping(path = "/welcome-bean/{name}")
    public HelloWorldBean getMessagePath(@PathVariable String name) {

        return new HelloWorldBean(String.format("Welcome to spring boot
%s", name));
    }
}
```

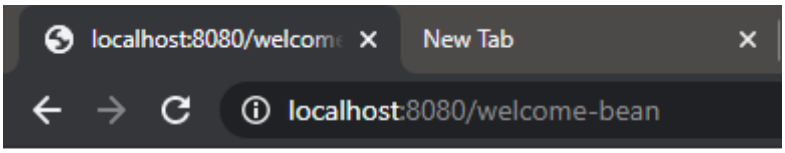
HelloWorldBean

```
public class HelloWorldBean {
    private String message;

    public HelloWorldBean(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    @Override
    public String toString() {
        return "HelloWorldBean{" +
                "message='" + message + '\'' +
                '}';
    }
}
```

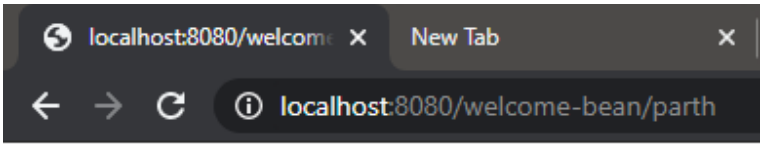
Output:



Welcome to spring boot



{"message":"Welcome to spring boot"}



{"message":"Welcome to spring boot parth"}

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

  Employee Bean:-

```
public class Employee {

    private Integer id;

    @Positive(message = "Age must be a positive integer")
    private Integer age;

    @Size(min = 3, message = "Name should have at least 3 characters")
    private String name;
```

```
    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 Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", age=" + age +
                ", name='" + name + '\'' +
                '}';
    }
}
```

Services:-

```
@Component
public class EmployeeDaoService {

    private static List<Employee> employeeList = new
ArrayList<Employee>();
    private static int empCount = 5;

    static {
        employeeList.add(new Employee(1, "Shubham", 24));
        employeeList.add(new Employee(2, "Vardan", 26));
        employeeList.add(new Employee(3, "Abhay", 25));
        employeeList.add(new Employee(4, "Rishabh", 23));
        employeeList.add(new Employee(5, "Vikas", 25));

    }

    public List<Employee> getAllEmployeeList() {
        return employeeList;
```

```
    }

    public Employee createEmployee(Employee employee) {
        if(employee.getId() == null)
        {
            employee.setId(++empCount);
        }
        employeeList.add(employee);
        return employee;
    }

    public Employee findOneEmployee(int id)
    {
        for(Employee emp : employeeList)
        {
            if(emp.getId() == id)
            {
                return emp;
            }
        }
        return null;
    }

    public Employee deleteEmployeeById(int id)
    {
        Iterator<Employee> iterator = employeeList.iterator();
        while(iterator.hasNext())
        {
            Employee emp = iterator.next();
            if(emp.getId() == id)
            {
                iterator.remove();
                return emp;
            }
        }

        return null;
    }

    public Employee updateEmployee(Employee employee, int id)
    {
        for (Employee emp : employeeList) {
            if (emp.getId() == id) {
                employeeList.set(employeeList.indexOf(emp), employee);
                return emp;
            }
        }

        return null;
    }
}
```

- Implement GET http request for Employee to get list of employees.

  Full Implementation of the resource class is on github(EmployeeResource.java)

```
//Get All Employees
@GetMapping(path = "/all-employee")
public List<Employee> reteriveAllEmployee() {
```

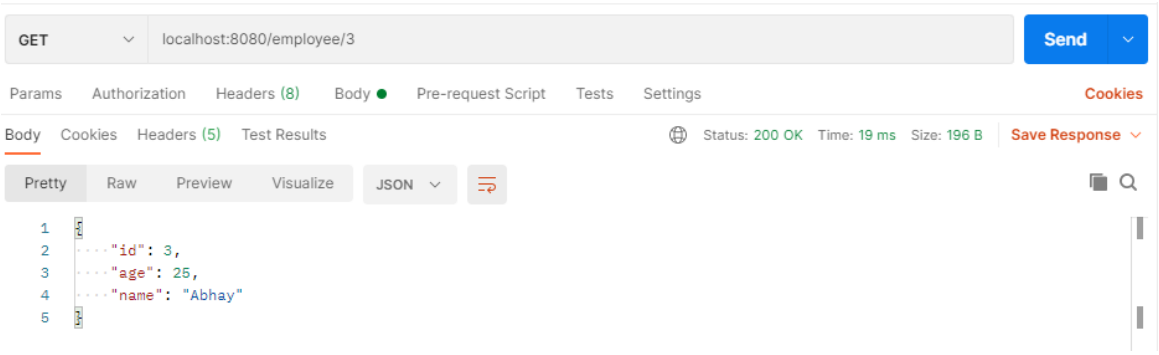
```
    return employeeDaoService.getAllEmployeeList();
}
```

Output:

- Implement GET http request using path variable to get one employee

```
//Get One Employee
@GetMapping(path = "/employee/{id}")
public Employee reteriveOneEmployee(@PathVariable int id) {
    Employee emp1 = employeeDaoService.findOneEmployee(id);
    if (emp1 == null)
        throw new EmployeeNotFoundException("Employee with id: " +
id + " not found !!");
    return emp1;
}
```

Output:

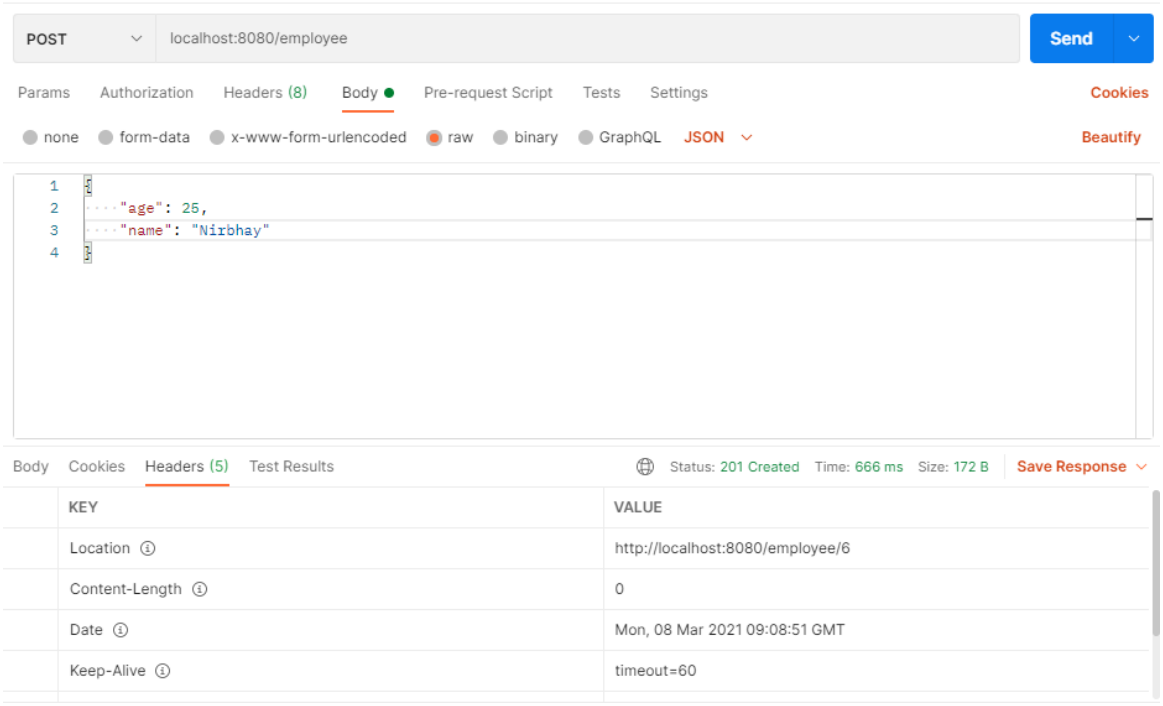


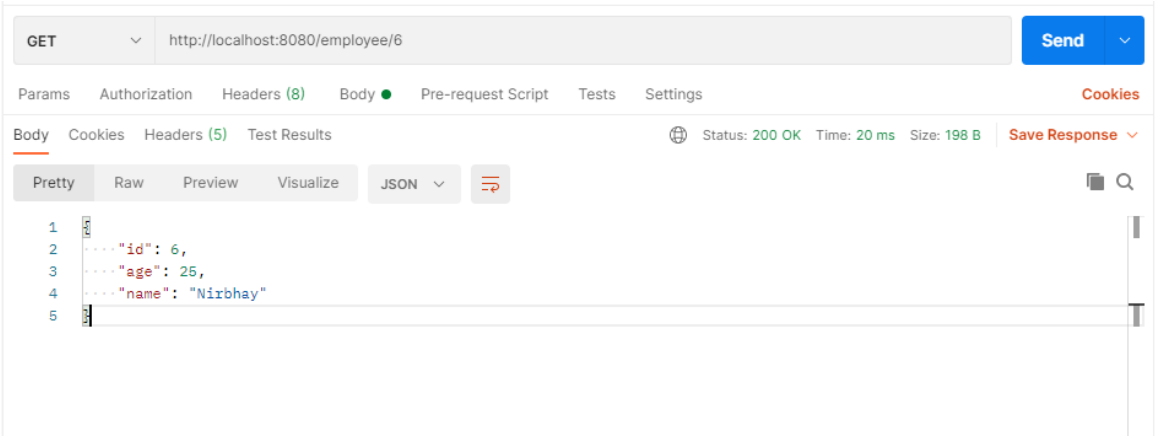
- Implement POST http request for Employee to create a new employee.

```
//Create a new user
@PostMapping(path = "/employee")
public ResponseEntity<Object> addEmployee(@RequestBody Employee
employee) {
    Employee employee1 =
employeeDaoService.createEmployee(employee);
    URI location = ServletUriComponentsBuilder
            .fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(employee1.getId()).toUri();
```

```
        return ResponseEntity.created(location).build();
    }
```

Output





- Implement Exception Handling for resource not found

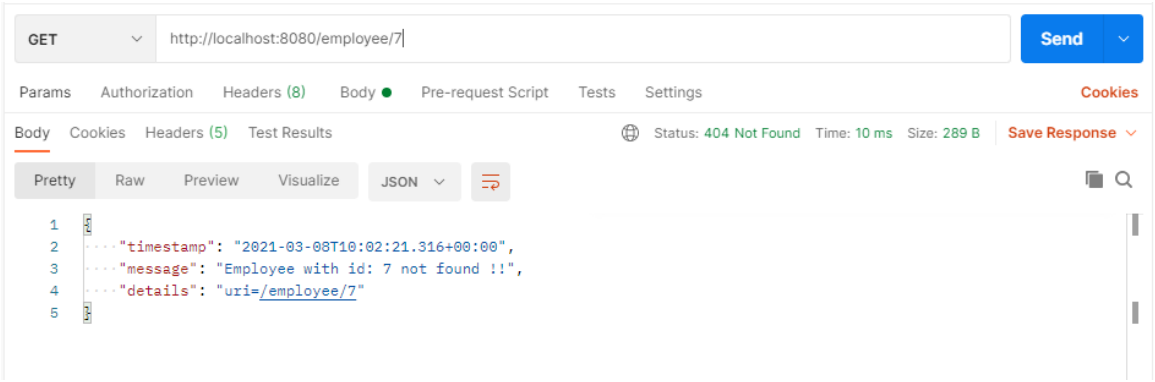
```
public class ExceptionResponse{
    private Date timestamp;
    private String message, details;

    public 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;
    }

}
```

CustomizedResponseException.Java

```
@ControllerAdvice
@RestController
public class CustomizedResponseException  extends
ResponseEntityExceptionHandler {

    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object> handleAllException(Exception ex,
WebRequest request)
    {
        ExceptionResponse exceptionResponse = new ExceptionResponse(new
Date(), ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<Object>(exceptionResponse,
HttpStatus.INTERNAL_SERVER_ERROR);
    }


    @ExceptionHandler(EmployeeNotFoundException.class)
    public final ResponseEntity<Object>
handleUserNotFoundException(EmployeeNotFoundException ex, WebRequest
request)
    {
        ExceptionResponse exceptionResponse = new ExceptionResponse(new
Date(), ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<Object>(exceptionResponse,
HttpStatus.NOT_FOUND);
    }

    @Override
    protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
HttpHeaders headers, HttpStatus status, WebRequest request) {
        ExceptionResponse exceptionResponse = new ExceptionResponse(new
```

```
Date(), "Validation Failed", ex.getBindingResult().toString());
        return new ResponseEntity<Object>(exceptionResponse,
HttpStatus.BAD_REQUEST);
    }

}
```

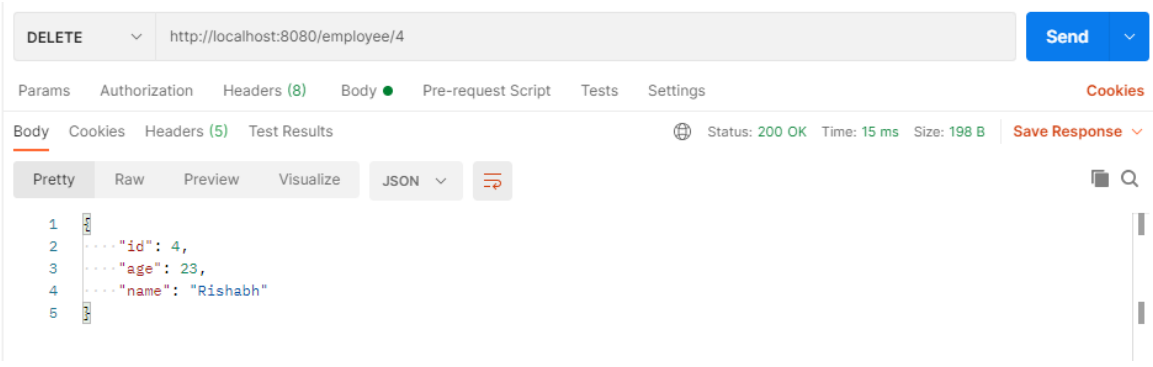
Output:-

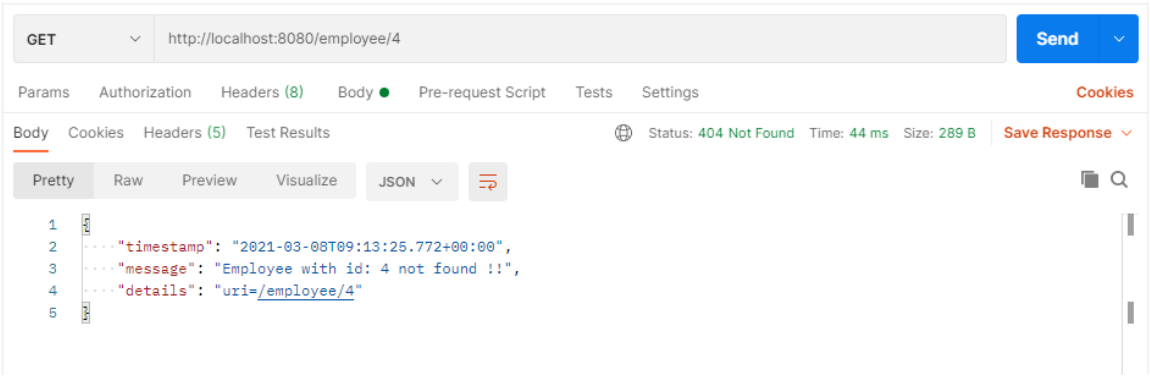


- Implement DELETE http request for Employee to delete employee

```
//Delete an Employee
@DeleteMapping(path = "/employee/{id}")
public Employee deleteOneEmployee(@PathVariable int id) {
    Employee emp1 = employeeDaoService.deleteEmployeeById(id);
    if (emp1 == null)
        throw new EmployeeNotFoundException("Employee with id: " +
id + " not found !!");
    return emp1;
}
```

Output:-

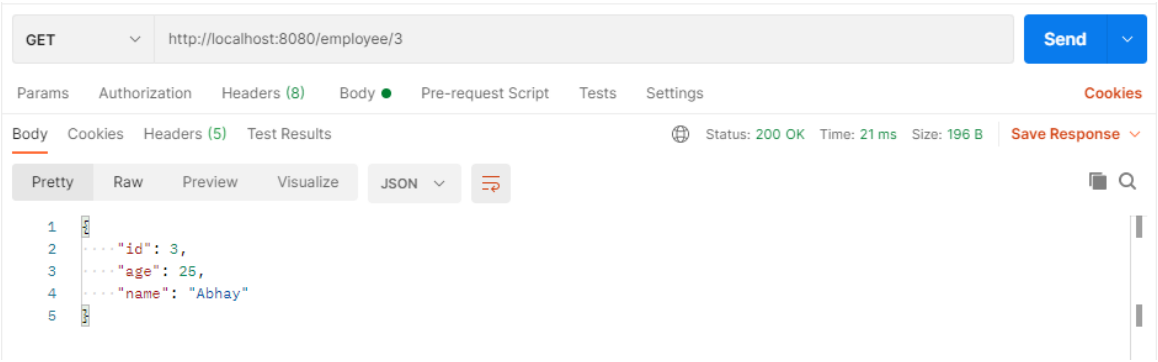


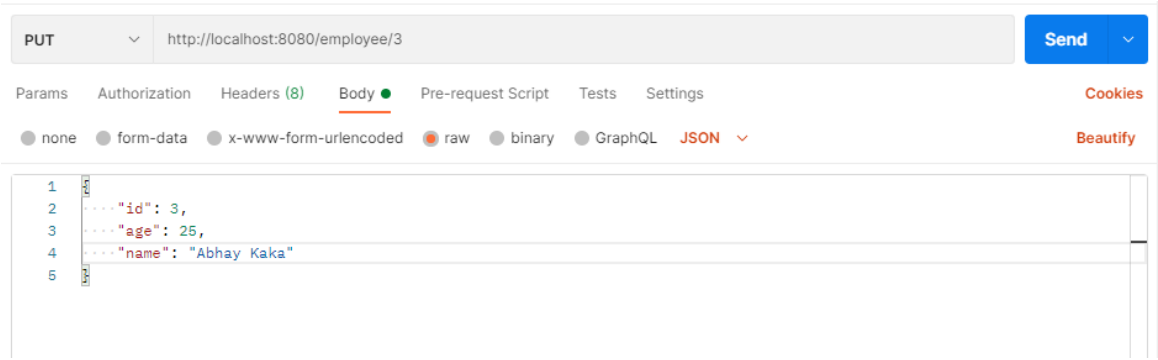


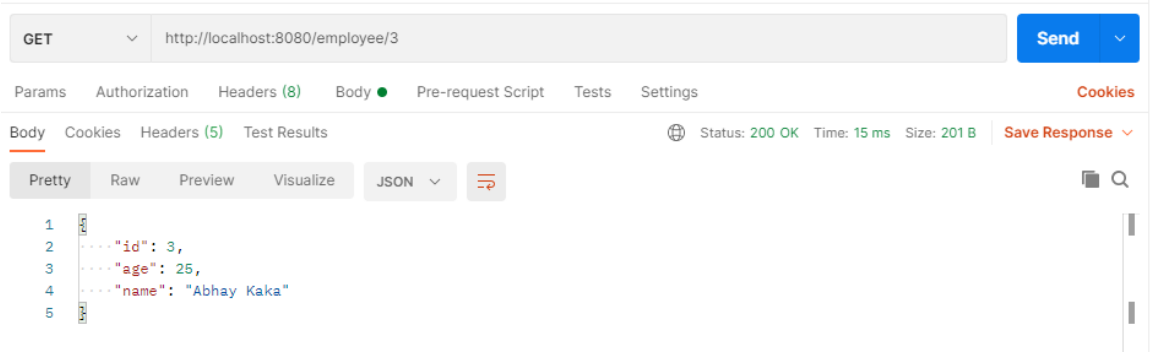
- Implement PUT http request for Employee to update employee

```
//Updates an Employee
@PutMapping(path = "/employee/{id}")
public Employee updateOneEmployee(@RequestBody Employee employee,
@PathVariable int id) {
    Employee emp1 =
employeeDaoService.updateEmployee(employee,id);
    if (emp1 == null)
        throw new EmployeeNotFoundException("Employee with id: " +
id + " not found !!");
    return emp1;
}
```

Output:-





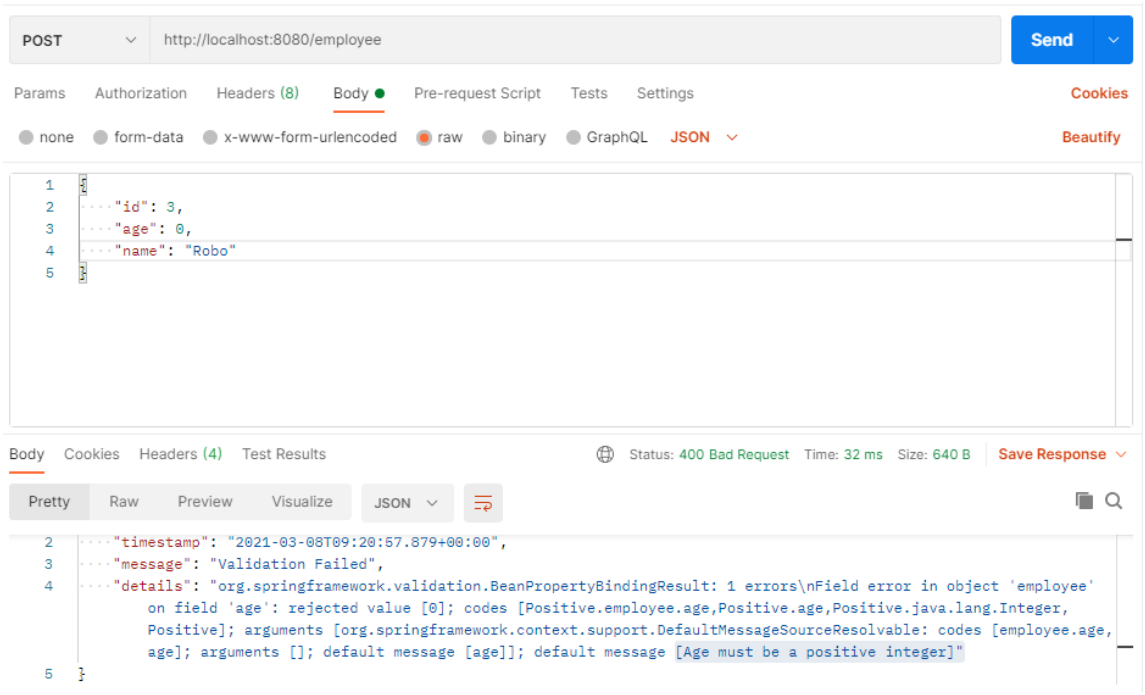


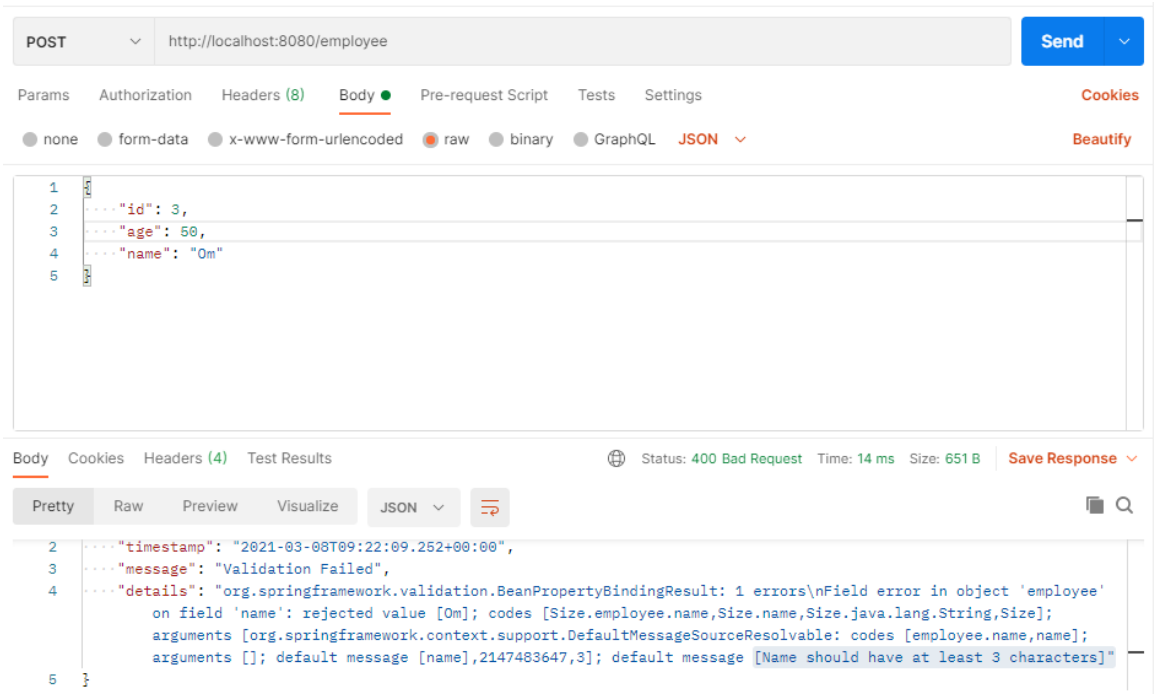
- Apply validation while create a new employee using POST http Request.

```
//Create a new user
@PostMapping(path = "/employee")
public ResponseEntity<Object> addEmployee(@Valid @RequestBody
Employee employee) {
    Employee employee1 =
employeeDaoService.createEmployee(employee);
    URI location = ServletUriComponentsBuilder
            .fromCurrentRequest()
            .path("/{id}")
            .buildAndExpand(employee1.getId()).toUri();

    return ResponseEntity.created(location).build();
}
```

```
public class Employee {

    private Integer id;

    @Positive(message = "Age must be a positive integer")
    private Integer age;

    @Size(min = 3, message = "Name should have at least 3 characters")
    private String name;
```

Output:-


POST
http://localhost:8080/employee
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 3,
"age": 0,
"name": "Robo"
}
Body
Cookies
Headers (4)
Test Results
Status: 400 Bad Request
Time: 32 ms
Size: 640 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
"timestamp": "2021-03-08T09:20:57.879+00:00",
"message": "Validation Failed",
"details": "org.springframework.validation.BeanPropertyBindingResult: 1 errors\nField error in object 'employee' on field 'age': rejected value [0]; codes [Positive.employee.age,Positive.age,Positive.java.lang.Integer,Positive]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [employee.age,age]; arguments []; default message [age]]; default message [Age must be a positive integer]"
}


POST
http://localhost:8080/employee
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 3,
"age": 50,
"name": "Om"
}
Body
Cookies
Headers (4)
Test Results
Status: 400 Bad Request
Time: 14 ms
Size: 651 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
"timestamp": "2021-03-08T09:22:09.252+00:00",
"message": "Validation Failed",
"details": "org.springframework.validation.BeanPropertyBindingResult: 1 errors\nField error in object 'employee' on field 'name': rejected value [Om]; codes [Size.employee.name,Size.name,Size.java.lang.String,Size];
arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [employee.name,name];
arguments []; default message [name],2147483647,3]; default message [Name should have at least 3 characters]"
}

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