

# Assignment - RestFul Web Services - Part 1

Q1) Create a simple REST ful service in Spring Boot which returns the Response "Welcome to spring boot".

```
1  package com.akash.restful_part_1.controller;
2
3
4  import com.akash.restful_part_1.service.MyService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.web.bind.annotation.GetMapping;
7  import org.springframework.web.bind.annotation.RequestMapping;
8  import org.springframework.web.bind.annotation.RestController;
9
10 @RestController
11 @RequestMapping("/api/rest")
12 public class MyController {
13
14     @Autowired
15     private MyService myService;
16
17     @GetMapping("/welcome")
18     public String welcome() {
19         return myService.welcome();
20     }
21 }
22
```

```
1 package com.akash.restful_part_1.service;
2
3 import org.springframework.stereotype.Service;
4
5 @Service
6 public class MyService {
7     public String welcome(){ no usages
8         return "Welcome to spring boot";
9     }
10 }
11
```

JVM / New Folder / Welcome route

GET localhost:8080/api/rest/welcome Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

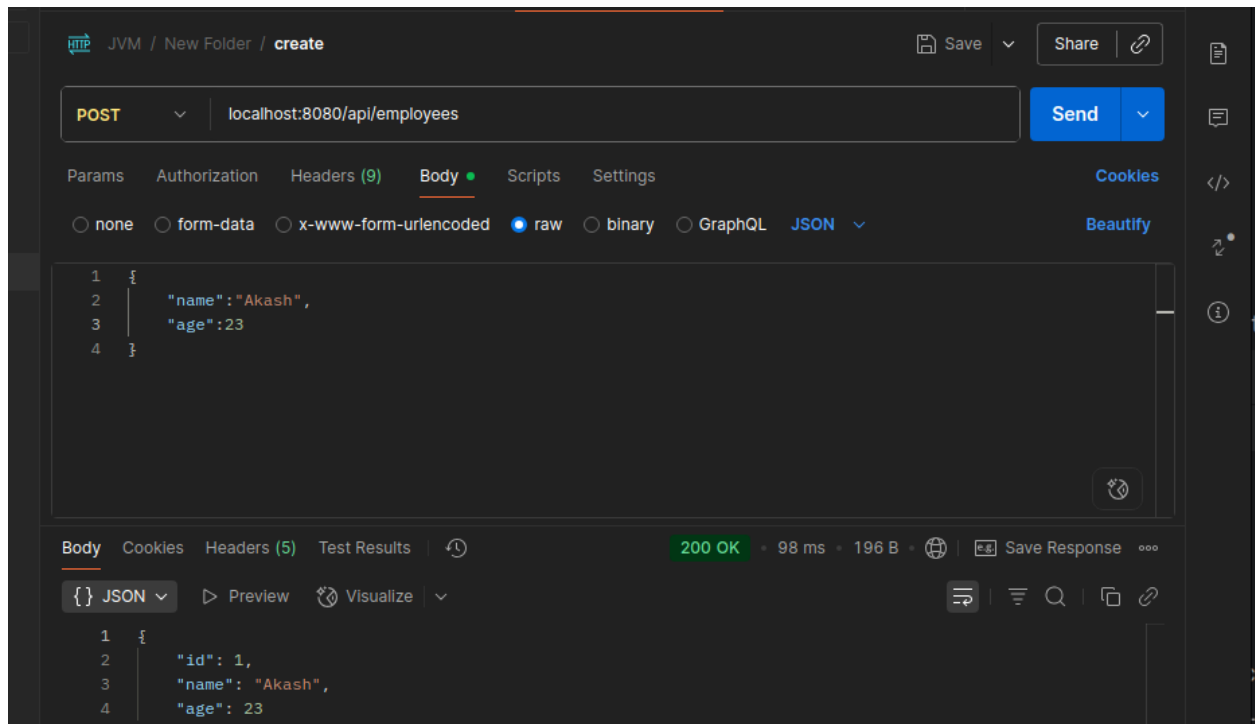
	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 48 ms • 186 B • Save Response

Raw Preview Visualize

1 Welcome to spring boot

Q2) Create an Employee Bean(id, name, age) and service to perform different operations related to employee.



```
@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    @Autowired
    private EmployeeService service;

    @GetMapping
    public List<Employee> getAllEmployees() {
        return service.getAllEmployees();
    }

    @GetMapping("/{id}")
    public Employee getEmployeeById(@PathVariable Long id) {
        return service.getEmployeeById(id);
    }

    @PostMapping
    public Employee createEmployee(@Valid @RequestBody Employee employee) {
        return service.createEmployee(employee);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteEmployee(@PathVariable Long id) {
        service.deleteEmployee(id);
    }
}
```

```
import java.util.Map;

@Service
public class EmployeeService {

    private Map<Long, Employee> employees = new HashMap<>();
    private long customId = 1L;

    private synchronized Long generateCustomId() {
        return customId++;
    }

    public List<Employee> getAllEmployees() {
        return new ArrayList<>(employees.values());
    }

    public Employee getEmployeeById(Long id) {
        Employee emp = employees.get(id);
        if (emp == null) throw new ResourceNotFoundException("Employee not found");
        return emp;
    }

    public Employee createEmployee(Employee employee) {
        employee.setId(generateCustomId());
        employees.put(employee.getId(), employee);
        return employee;
    }
}
```

```
package com.akash.restful_part_1.model;

import org.springframework.stereotype.Component;

@Component 16 usages
public class Employee {
    private Long id; 2 usages
    private String name; 2 usages
    private int age; 2 usages

    public Long getId() { 1 usage
        return id;
    }

    public void setId(Long id) { 2 usages
        this.id = id;
    }

    public String getName() { no usages
```

Q3) Implement GET http request for Employee to get list of employees.

```
public List<Employee> getAllEmployees() { no usages
    return new ArrayList<>(employees.values());
}
```

```
@GetMapping
public List<Employee> getAllEmployees() {
    return service.getAllEmployees();
}
```

The screenshot shows an HTTP client interface with the following details:

- URL:** `localhost:8080/api/employees`
- Method:** `GET`
- Status:** `200 OK` (6 ms, 231 B)
- Response Body (JSON):**

```
[
  {
    "id": 1,
    "name": "Akash",
    "age": 23
  },
  {
    "id": 2,
    "name": "Sahil",
    "age": 23
  }
]
```

Q4) Implement GET http request using path variable to get one employee

```
@GetMapping("/{id}")
public Employee getEmployeeById(@PathVariable Long id) {
    return service.getEmployeeById(id);
}
```

```
public Employee getEmployeeById(Long id) { no usages
    Employee emp = employees.get(id);
    if (emp == null) throw new ResourceNotFoundException("Employee not found with id: " + id);
    return emp;
}
```

The screenshot shows a REST client interface with the following details:

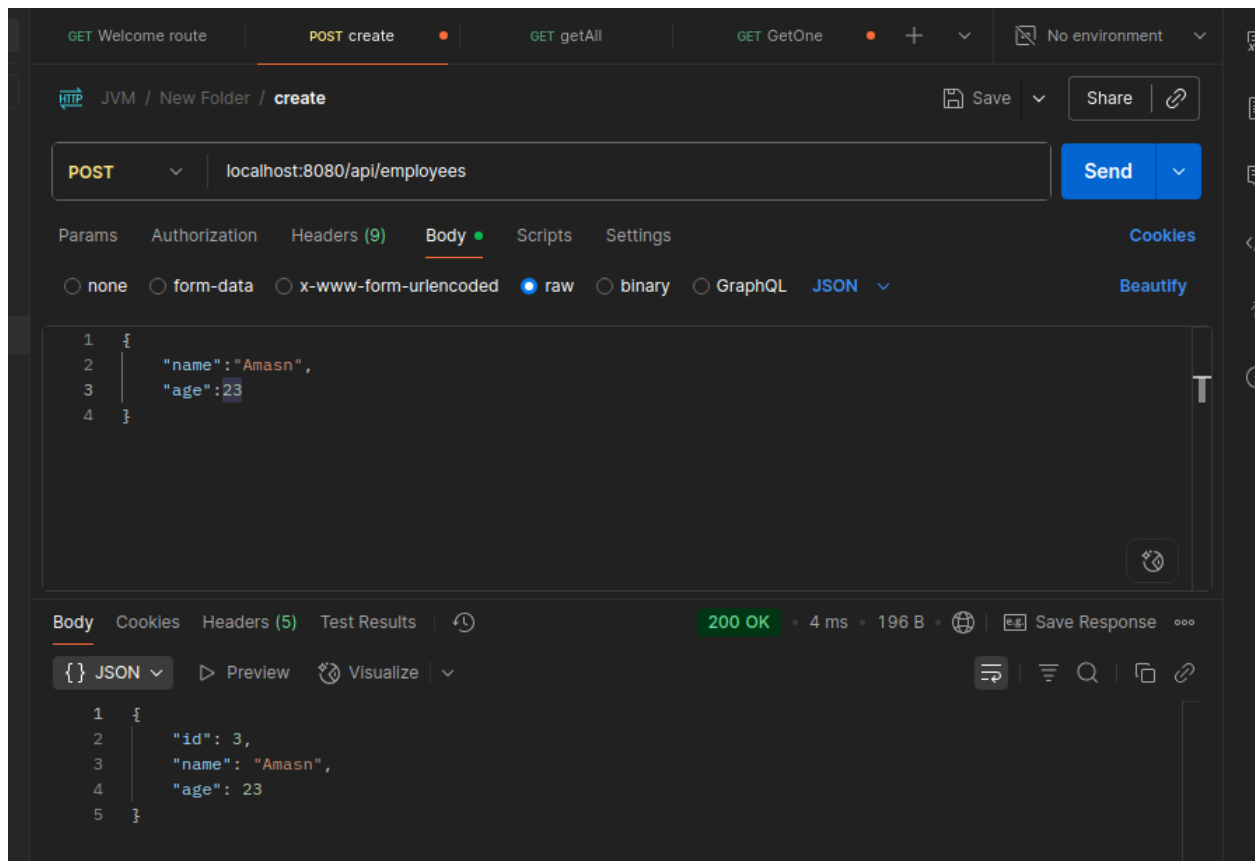
- Method:** GET
- URL:** localhost:8080/api/employees/1
- Status:** 200 OK
- Response Time:** 5 ms
- Response Size:** 196 B
- Response Body (JSON):**

```
{
  "id": 1,
  "name": "Akash",
  "age": 23
}
```

Key	Value	Description
Key	Value	Description



Q5) Implement POST http request for Employee to create a new employee.



```
31  
32 @      public Employee createEmployee(Employee employee) { no usages  
33         employee.setId(generateCustomId());  
34         employees.put(employee.getId(), employee);  
35         return employee;  
36     }  
37
```

```
@PostMapping  
public Employee createEmployee(@Valid @RequestBody Employee employee) {  
    return service.createEmployee(employee);  
}
```

## Q6) Implement Exception Handling for resource not found

```
package com.akash.restful_part_1.exception;

public class ResourceNotFoundException extends RuntimeException { 4 usages
    public ResourceNotFoundException(String message) { 3 usages
        super(message);
    }
}
```

```
com.akash.restful_part_1.exception;

org.springframework.http.HttpStatus;
org.springframework.http.ResponseEntity;
org.springframework.web.bind.MethodArgumentNotValidException;
org.springframework.web.bind.annotation.ExceptionHandler;
org.springframework.web.bind.annotation.RestControllerAdvice;

ntrollerAdvice
class GlobalExceptionHandler {

ceptionHandler(ResourceNotFoundException.class)
@lic ResponseEntity<String> handleResourceNotFoundException(ResourceNotFoundException e) {
    return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
}
```

Q7) Implement DELETE http request for Employee to delete employee

```
@DeleteMapping("/{id}")
public ResponseEntity<String> deleteEmployee(@PathVariable Long id) {
    service.deleteEmployee(id);
    return ResponseEntity.ok(body: "Employee deleted successfully");
}
```

```
public void deleteEmployee(Long id) { no usages
    if (!employees.containsKey(id)) throw new ResourceNotFoundException("Employee not found with id: " + id)
    employees.remove(id);
}
```

The screenshot shows the Postman interface for a DELETE request. The request is sent to `localhost:8080/apl/employees/3` and returns a `200 OK` status with a response body of `Employee deleted successfully`.

Request Details:

- Method: DELETE
- URL: `localhost:8080/apl/employees/3`
- Environment: No environment

Response Details:

- Status: 200 OK
- Time: 5 ms
- Size: 193 B
- Body: Employee deleted successfully

Key	Value	Description
Key	Value	Description

## Q8) Implement PUT http request for Employee to update employee

```
public Employee updateEmployee(Long id, Employee updated) {  
    no usages  
    if (!employees.containsKey(id)) throw new ResourceNotFoundException("Employee not found with id: " + id);  
    updated.setId(id);  
    employees.put(id, updated);  
    return updated;  
}
```

```
}  
  
@PutMapping("/{id}")  
public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee updatedEmployee) {  
    return service.updateEmployee(id, updatedEmployee);  
}
```

The screenshot displays a REST client interface within an IDE. The top bar shows various HTTP methods: GET Welcome, POST create, GET getAll, GET GetOne, DEL Delete, and PUT Update (selected). The URL bar shows the endpoint `localhost:8080/api/employees/2`. The request method is set to PUT. The request body is a JSON object: `{ "name": "Sahil", "age": 56 }`. The response status is 200 OK, with a response time of 4 ms and a body size of 196 B. The response body is a JSON object: `{ "id": 2, "name": "Sahil", "age": 56 }`.

Q9)Apply validation while create a new employee using POST http Request.

```
@PostMapping
public Employee createEmployee(@Valid @RequestBody Employee employee) {
    return service.createEmployee(employee);
}
```

```
6
7 @Component 16 usages
8 public class Employee {
9     private Long id; 3 usages
10
11     @NotBlank(message = "Name is mandatory") 3 usages
12     private String name;
13
14     @Min(value = 18, message = "Age must be >= 18") 3 usages
15     private int age;
16
17     public Employee() {}
18
19     public Employee(Long id, String name, int age) {
20         this.id = id;
21         this.name = name;
22         this.age = age;
23     }
24
```

JVM / New Folder / create

Save

Share

POST

localhost:8080/api/employees

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   |
3   |   "age":23
4   | }
```

Body

Cookies

Headers (4)

Test Results

400 Bad Request

102 ms

161 B

Save Response

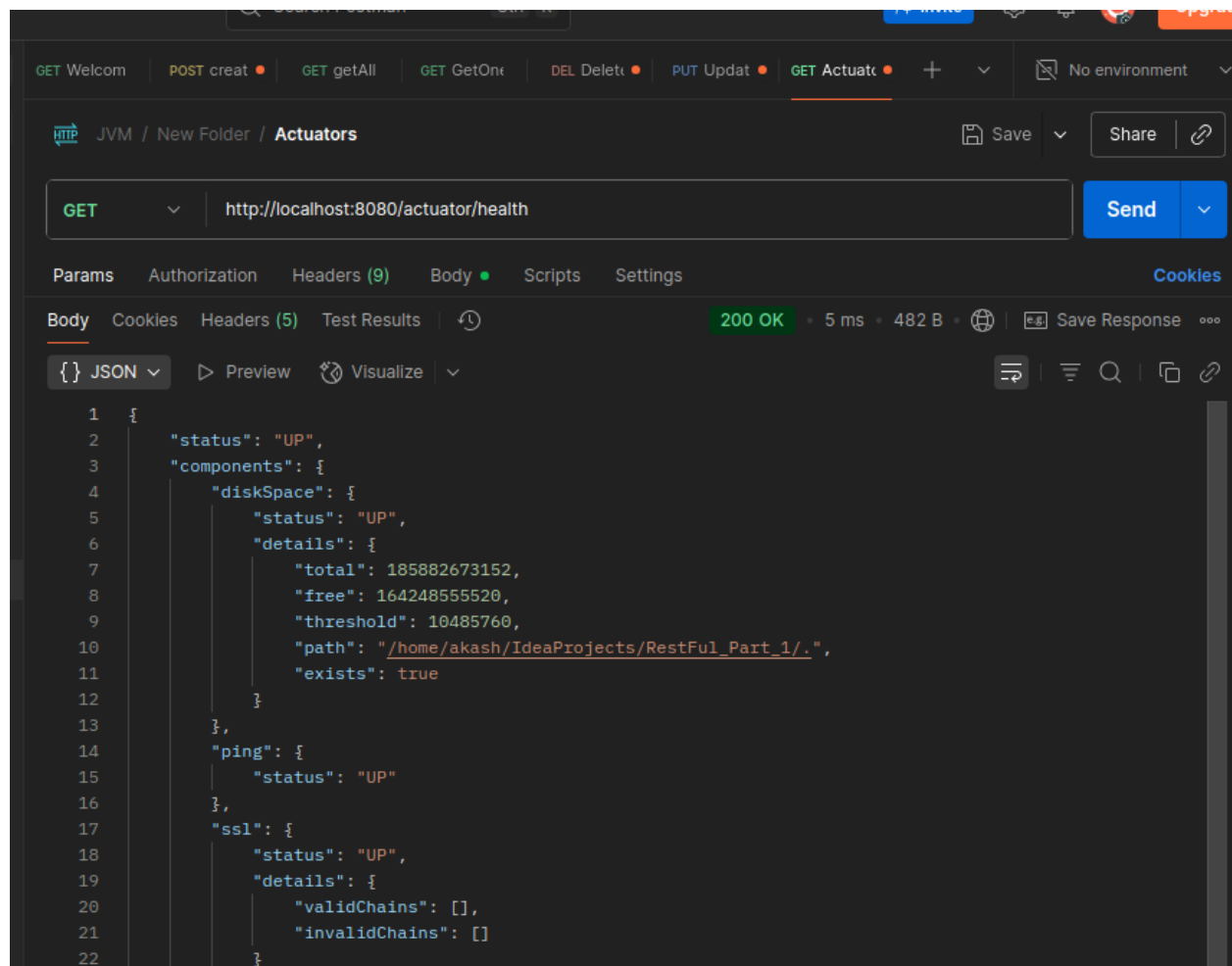
Raw

Preview

Visualize

```
1 Name is mandatory
```

Q10) Configure actuator in your project to check the health of application and get the information about various beans configured in your application



GET Welcom POST creat GET getAll GET GetOnr DEL Delet PUT Updat GET Actuato + No environment

JVM / New Folder / Actuators Save Share

GET http://localhost:8080/actuator/beans Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 8 ms 113.7 KB Save Response

{ } JSON Preview Visualize

```
1 {
2   "contexts": {
3     "RestFul_Part_1": {
4       "beans": {
5         "endpointCachingOperationInvokerAdvisor": {
6           "aliases": [],
7           "scope": "singleton",
8           "type": "org.springframework.boot.actuate.endpoint.invoker.cache.
          CachingOperationInvokerAdvisor",
9           "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/
          endpoint/EndpointAutoConfiguration.class]",
10          "dependencies": [
11            "org.springframework.boot.actuate.autoconfigure.endpoint.
              EndpointAutoConfiguration",
12            "environment"
13          ]
14        },
15        "defaultServletHandlerMapping": {
16          "aliases": [],
17          "scope": "singleton",
18          "type": "org.springframework.web.servlet.HandlerMapping",
19          "resource": "class path resource [org/springframework/boot/autoconfigure/web/
          servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",
20          "dependencies": [
21            "org.springframework.boot.autoconfigure.web.servlet"
```

Postbot Runner Capture requests Cookies Vault Trash



Save Copy Collapse All Expand All (slow) Filter JSON

▼ contexts:

▼ RestFul\_Part\_1:

▼ beans:

- ▶ endpointCachingOperationInvokerAdvisor:
- ▶ defaultServletHandlerMapping:
- ▶ globalExceptionHandler:
- ▶ applicationTaskExecutor:
- ▶ observationRegistryPostProcessor:
- ▶ characterEncodingFilter:
- ▶ forceAutoProxyCreatorToUseClassProxying:
- ▶ healthEndpointGroups:
- ▶ management.endpoint.health-org.springframework.boot.actuate.autoconfigure.health.HealthEndpointProperties:
- ▶ webEndpointDiscoverer:
- ▶ org.springframework.boot.autoconfigure.web.servlet.MultipartAutoConfiguration:
- ▶ org.springframework.boot.autoconfigure.http.client.HttpClientAutoConfiguration:
- ▶ org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration\$DispatcherServletRegistrationConf
- ▶ preserveErrorControllerTargetClassPostProcessor:
- ▶ org.springframework.boot.autoconfigure.jackson.JacksonAutoConfiguration\$JacksonObjectMapperBuilderConfiguration:
- ▶ logbackMetrics:
- ▶ management.info-org.springframework.boot.actuate.autoconfigure.info.InfoContributorProperties:
- ▶ org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration:
- ▶ webEndpointPathMapper:
- ▶ org.springframework.boot.actuate.autoconfigure.cache.CachesEndpointAutoConfiguration:
- ▶ propertySourcesPlaceholderConfigurer:
- ▶ org.springframework.boot.actuate.autoconfigure.endpoint.jmx.JmxEndpointAutoConfiguration:

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

EmployeeService.java EmployeeController.java Employee.java pom.xml (RestFul\_Part\_1) application.properties x

```
spring.application.name=RestFul_Part_1
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
```