

Spring boot request and response examples without database.

We will be seeing some real time API without database in this assignment.

Please refer to the json tutorial from below link.

<http://javabykiran.in/springboot/SpringBoot-JSON-JBKADVSPRINGBOOT3002.pdf>

javabyKiran
java | selenium | python

Question #1:

- Design a rest API where input and output will be as below.
 - Only need to use controller class.
 - No database needed.
 - Must use URL endpoint as provided.
 - Input – Nothing
 - URL - <http://localhost:8080/api/showallEmployee>
 - Output json—

```
[
  {
    "id": 1,
    "name": "KiranSir",
    "phoneno": "1234567890",
    "departmentit": "it",
    "status": "active",
    "createddtm": "2020-01-21 00:00:00.0",
    "createdby": "Vikash",
    "updateddtm": "2020-01-21 00:00:00.0",
    "updatedby": "Kumar",
    "cid": 1
  },
  {
    "id": 2,
    "name": "Suresh",
    "phoneno": "1234567890",
    "departmentit": "it",
    "status": "active",
    "createddtm": "2020-01-21 00:00:00.0",
    "createdby": "Suresh",
    "updateddtm": "2020-01-21 00:00:00.0",
    "updatedby": "Malhotra",
    "cid": 2
  }
]
```

Answer:

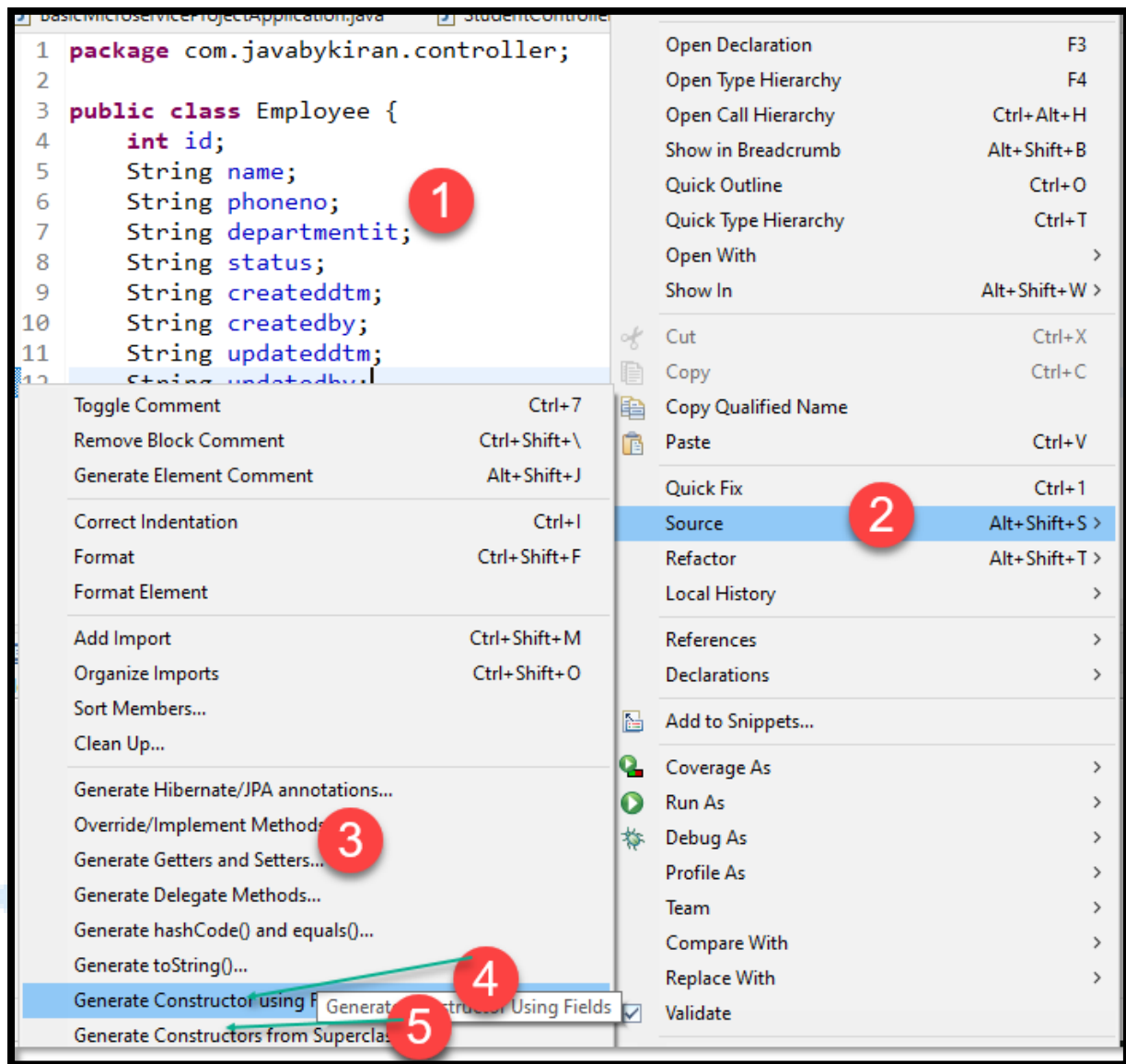
1. Analyze output its list of Employees.
2. We need to design pojo class Employee.
3. We need to mention all variables as mentioned in json in pojo class.
4. Getter and setter
5. Need to build logic in controller to return ArrayList<Employee>

Step#1

Create class Employee with variables. Values will be set from controller.

```
public class Employee {  
    int id;  
    String name;  
    String phoneno;  
    String departmentit;  
    String status;  
    String createddtm;  
    String createdby;  
    String updateddtm;  
    String updatedby;  
    int cid;  
}
```

Now add getters and setters and constructors in pojo class.



Code for class employee – only write variables other than this should be auto generated through eclipse source option

```
package com.javabykiran.controller;

public class Employee {
    int id;
    String name;
    String phoneno;
    String departmentit;
    String status;
    String createddtm;
    String createdby;
    String updateddtm;
    String updatedby;
```

```
public Employee() {
    super();
}
public Employee(int id, String name, String phoneno, String departmentit, String
status, String createddtm,
                String createdby, String updateddtm, String updatedby, int cid) {
    super();
    this.id = id;
    this.name = name;
    this.phoneno = phoneno;
    this.departmentit = departmentit;
    this.status = status;
    this.createddtm = createddtm;
    this.createdby = createdby;
    this.updateddtm = updateddtm;
    this.updatedby = updatedby;
    this.cid = cid;
}
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getPhoneno() {
    return phoneno;
}
public void setPhoneno(String phoneno) {
    this.phoneno = phoneno;
}
public String getDepartmentit() {
    return departmentit;
}
public void setDepartmentit(String departmentit) {
    this.departmentit = departmentit;
}
public String getStatus() {
    return status;
}
public void setStatus(String status) {
    this.status = status;
}
```

```
}  
public String getCreateddtm() {  
    return createddtm;  
}  
public void setCreateddtm(String createddtm) {  
    this.createddtm = createddtm;  
}  
public String getCreatedby() {  
    return createdby;  
}  
public void setCreatedby(String createdby) {  
    this.createdby = createdby;  
}  
public String getUpdateddtm() {  
    return updateddtm;  
}  
public void setUpdateddtm(String updateddtm) {  
    this.updateddtm = updateddtm;  
}  
public String getUpdatedby() {  
    return updatedby;  
}  
public void setUpdatedby(String updatedby) {  
    this.updatedby = updatedby;  
}  
public int getCid() {  
    return cid;  
}  
public void setCid(int cid) {  
    this.cid = cid;  
}  
int cid;  
}
```

Once this class is created, we need to start writing a controller
EmployeeController.java

```
package com.javabykiran.controller;  
  
import java.util.ArrayList;  
  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;
```

```
@RestController
public class EmployeeController {

    @RequestMapping("showallEmployee")
    public ArrayList<Employee> fetchAllEmployees () {

        ArrayList<Employee> al = new ArrayList<>();

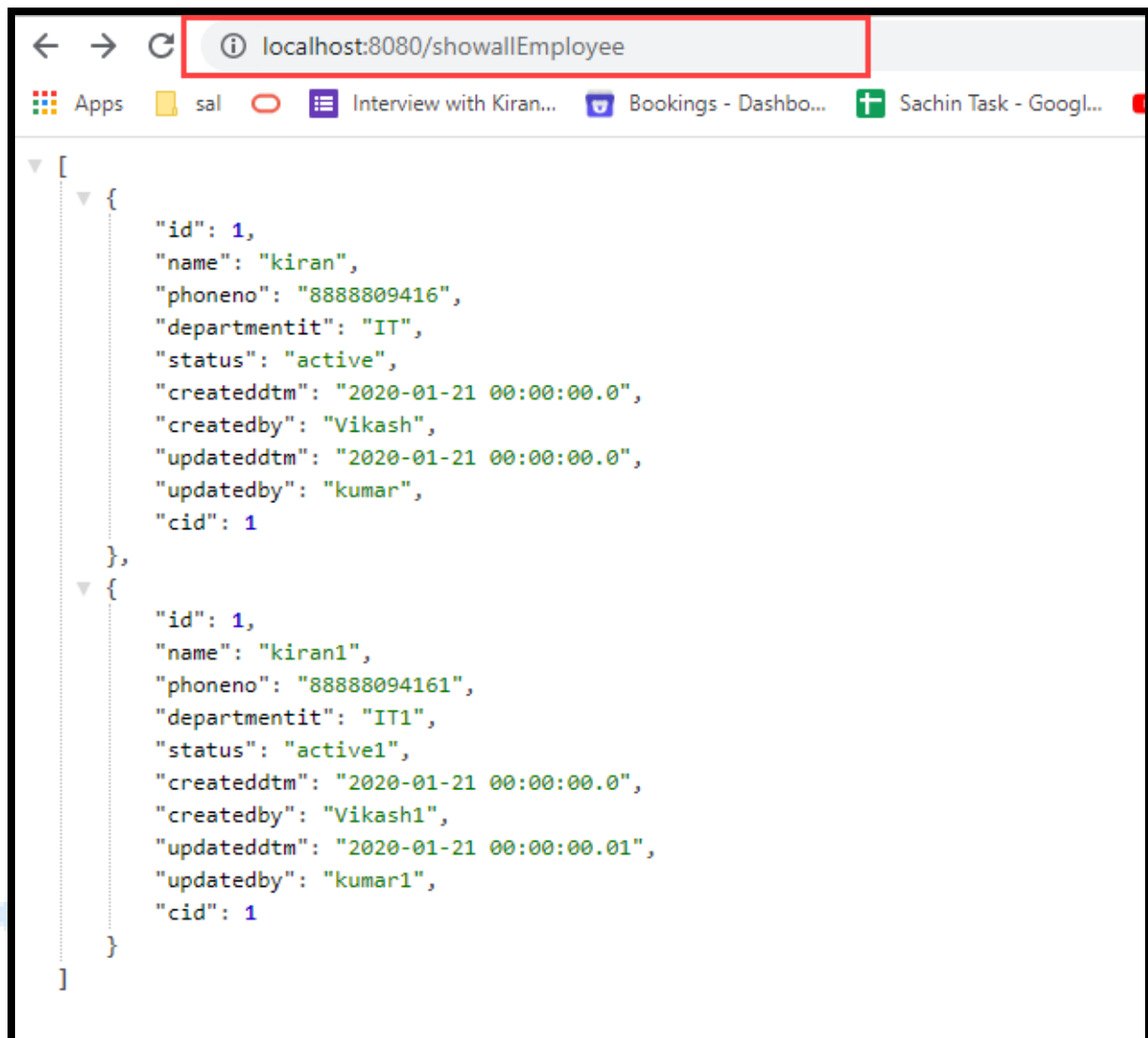
        Employee employee = new Employee(1, "kiran", "8888809416", "IT",
"active", "2020-01-21 00:00:00.0", "Vikash",
"2020-01-21 00:00:00.0", "kumar", 1);
        Employee employee1 = new Employee(1, "kiran1", "88888094161", "IT1",
"active1", "2020-01-21 00:00:00.0",
"Vikash1", "2020-01-21 00:00:00.01", "kumar1", 1);
        al.add(employee);
        al.add(employee1);
        return al;
    }
}
```

While running a project we need to start spring boot and hit url mentioned in address bar of a browser.

Always check console for any runtime errors if no errors found go to browser and hit url to see results

<http://localhost:8080/showallEmployee>

Now you can see on browser.



```
[
  {
    "id": 1,
    "name": "kiran",
    "phoneno": "8888809416",
    "departmentit": "IT",
    "status": "active",
    "createddtm": "2020-01-21 00:00:00.0",
    "createdby": "Vikash",
    "updateddtm": "2020-01-21 00:00:00.0",
    "updatedby": "kumar",
    "cid": 1
  },
  {
    "id": 1,
    "name": "kiran1",
    "phoneno": "88888094161",
    "departmentit": "IT1",
    "status": "active1",
    "createddtm": "2020-01-21 00:00:00.0",
    "createdby": "Vikash1",
    "updateddtm": "2020-01-21 00:00:00.01",
    "updatedby": "kumar1",
    "cid": 1
  }
]
```

Homework

- In above example use @GetMapping instead of @RequestMapping
- Solve below mentioned API on the same lines.

API #2

URL: http://localhost:8080/api/status/{status}

status >> ACTIVE AND INACTIVE – controller should have hard coded logic – if else

TYPE: Get

Input JSON - NO

Output JSON -

```
[  
  {  
    "id": 21,  
    "name": "Vikash",  
    "phoneno": "9743152782",  
    "departmentit": "it",  
    "status": "inactive",  
    "createddtm": "2020-01-23 13:12:20.0",  
    "createdby": "system",  
    "updateddtm": "2020-01-23 13:12:15.0",  
    "updatedby": "system",  
    "cid": 1  
  }  
]
```

API #3

URL : <http://localhost:8080/api/{eid}>

eid >> 1 or 2 or 3 – controller should have hard coded logic – if else

TYPE: Get

Input json - NO

Output json

```
[  
  {  
    "id": 1,  
    "name": "KiranSir",  
    "phoneno": "1234567890",  
    "departmentit": "it",  
    "status": "active",  
    "createddtm": "2020-01-21 00:00:00.0",  
    "createdby": "Vikash",  
    "updateddtm": "2020-01-21 00:00:00.0",  
    "updatedby": "Kumar",  
    "cid": 1  
  }  
]
```

API #4

URL : <http://localhost:8080/api/showEmployeesByName/{empname}>

ename >> jbk or kiran – controller should have hard coded logic – if else

TYPE: Get

Input json - NO

Output json

```
[  
  {  
    "id": 1,  
    "name": "KiranSir",  
    "phoneno": "1234567890",  
    "departmentit": "it",  
    "status": "active",  
    "createddtm": "2020-01-21 00:00:00.0",  
    "createdby": "Vikash",  
    "updateddtm": "2020-01-21 00:00:00.0",  
    "updatedby": "Kumar",  
    "cid": 1  
  }  
]
```

API #5

URL : : <http://localhost:8080/api/addemployee>

- In controller all data of input json should get printed. No database usage in this case.

TYPE: POST

Input json –

```
{  
    "name": "Shyam",  
    "phoneno": "897564123",  
    "departmentit": "Electrical",  
    "cid": 3  
}
```

Output json

No request code 200 OK

data is added successfully.

Tech specifications use below:

```
return new ResponseEntity ("Employee added in DB successfully", HttpStatus.OK);
```

in controller use below annotation

```
(@RequestBody User newUser) {
```

API #6

URL : : <http://localhost:8080/api/addemployee>

- In controller all data of input json should get printed. No database usage in this case.

TYPE: POST

Input json –

```
{  
    "name": "Shyam",  
    "phoneno": "897564123",  
    "departmentit": "Electrical",  
    "cid": 3  
}
```

Output json

No request code 200 OK
data is added successfully.

Tech specifications use below:

```
return new ResponseEntity ("Employee added in DB successfully", HttpStatus.OK);
```

API #7

URL : <http://localhost:8080/api/updatecountryname>

- In controller all data of input json should get printed. No database usage in this case.
- Try to get id in controller from input json and update country object with value provided through input json.

TYPE: PUT

Input json –

```
{  
    "cid": 4,  
    "cname": "UAE"  
}
```

Output json

No request code 200 OK

data is update successfully.

Tech specifications use below:

```
return new ResponseEntity ("Country updated in DB successfully", HttpStatus.OK);
```

API #8

URL: <http://localhost:8080/api/deletebycountryname/{countryname}>

Country name should be sent through URL.

- In controller all data of input json should get printed. No database usage in this case.
- Try to get countryname in controller from input json and update country object with value provided through input json.
- Return nothing only success full message.

TYPE: DELETE

Input json – NO

Output json

No request code 200 OK

data is update successfully.

API #9

URL: <http://localhost:8080/api/deleteemployeebyid/{eid}>

Eid will be sent through url.

- In controller all data of input json should get printed. No database usage in this case.
- Try to get eid in controller from input json and update country object with value provided through input json.
- Return nothing only success full message.

TYPE: DELETE

Input json –

```
{
  "cid": 4,
  "cname": "UAE"
}
```

Output json

No request code 200 OK

data is update successfully.

In controller use below annotation

getById(**@PathVariable** long id)

or

@PathVariable(value = "id") Long employeeId