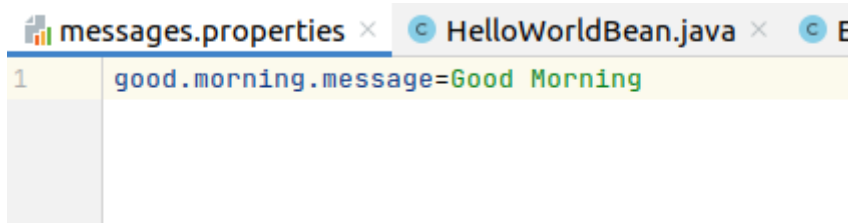


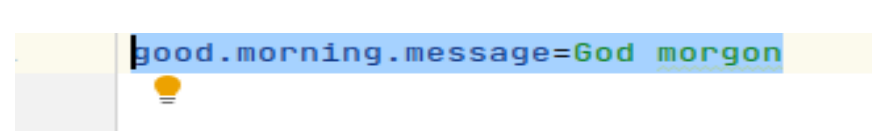
RestFul Web Service Part 2 Exercise

*Internationalization

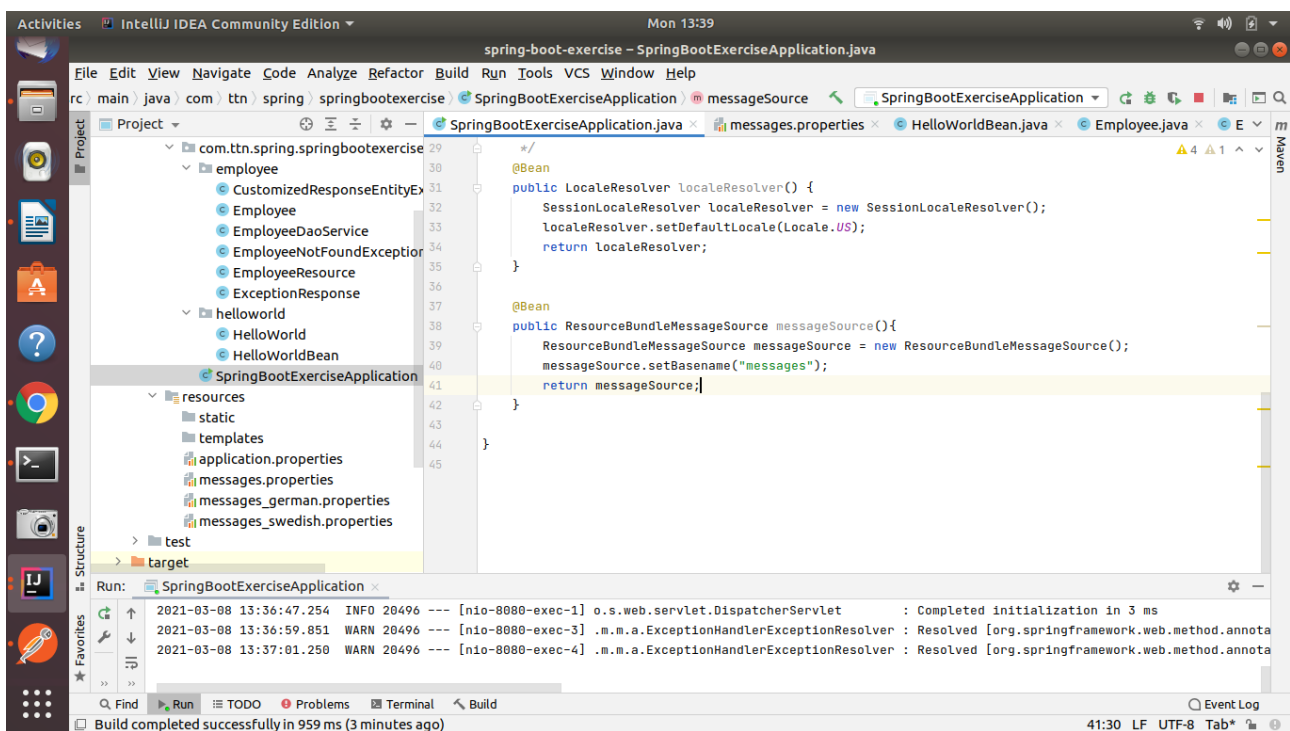
1. Add support for Internationalization in your application allowing messages to be shown in English, German and Swedish, keeping English as default.

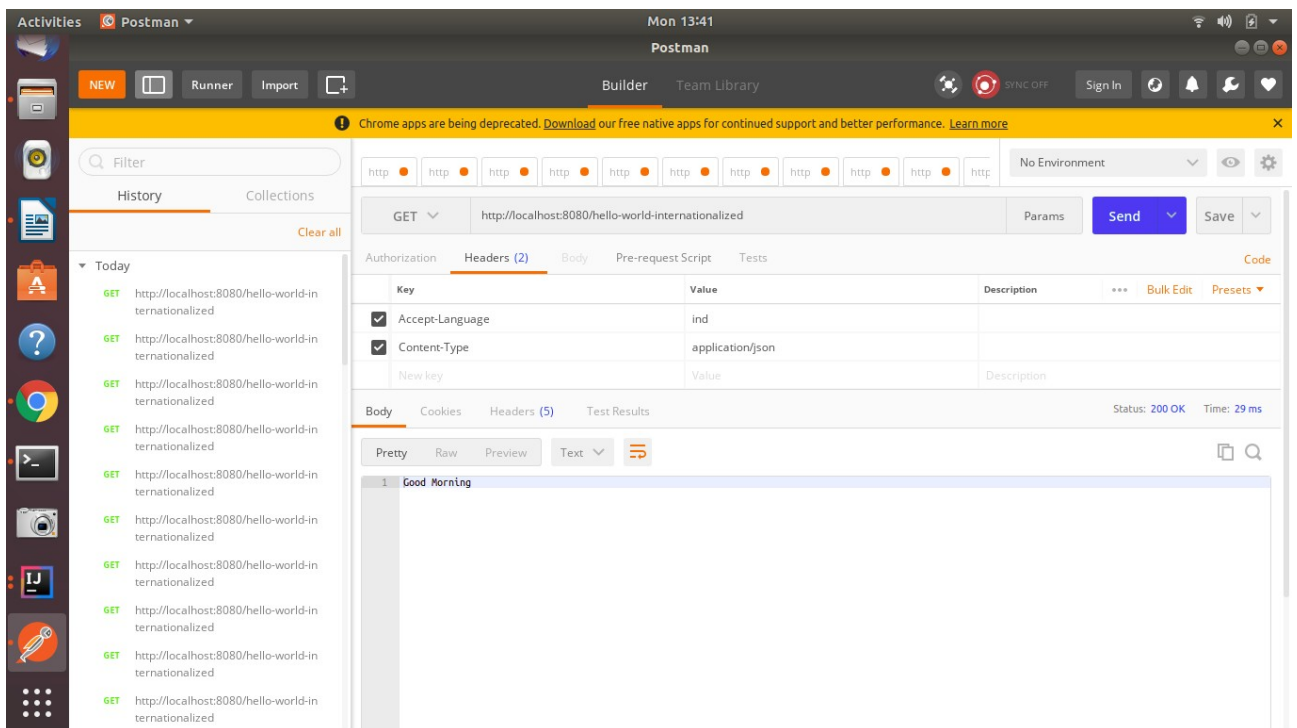
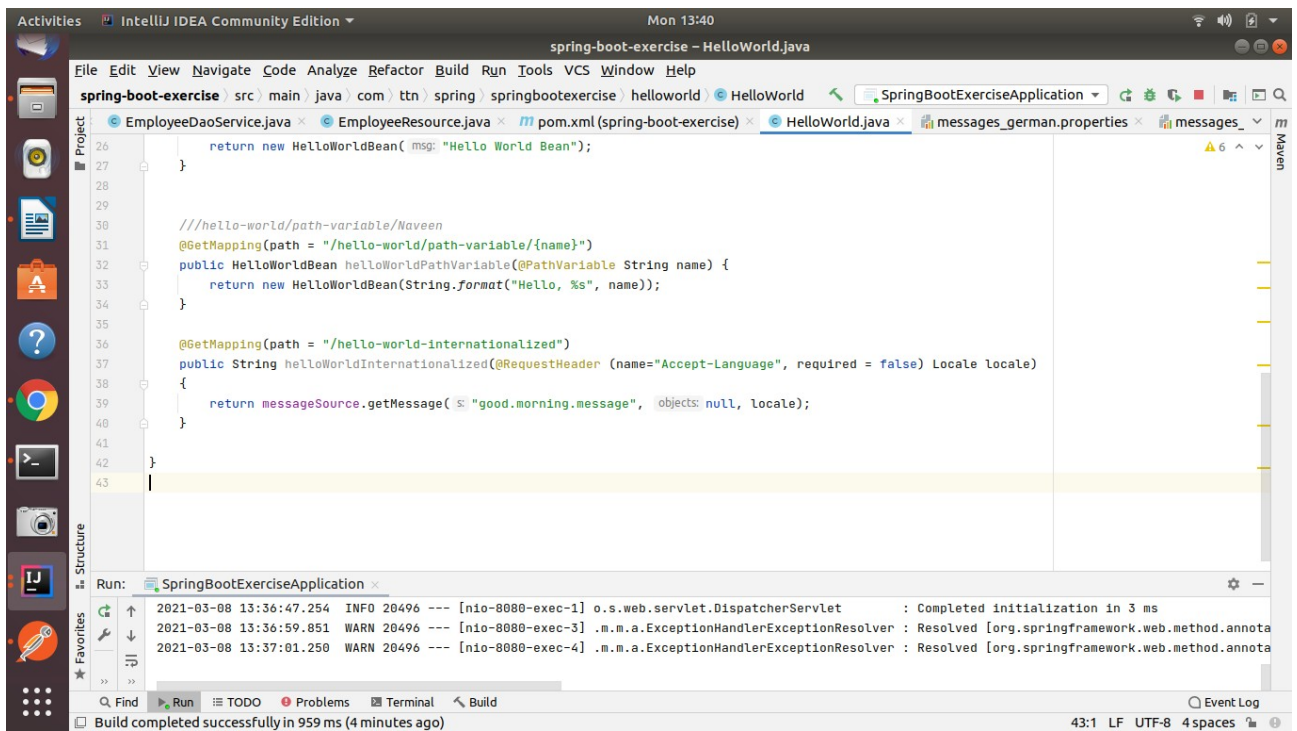


```
1 good.morning.message=Good Morning
```



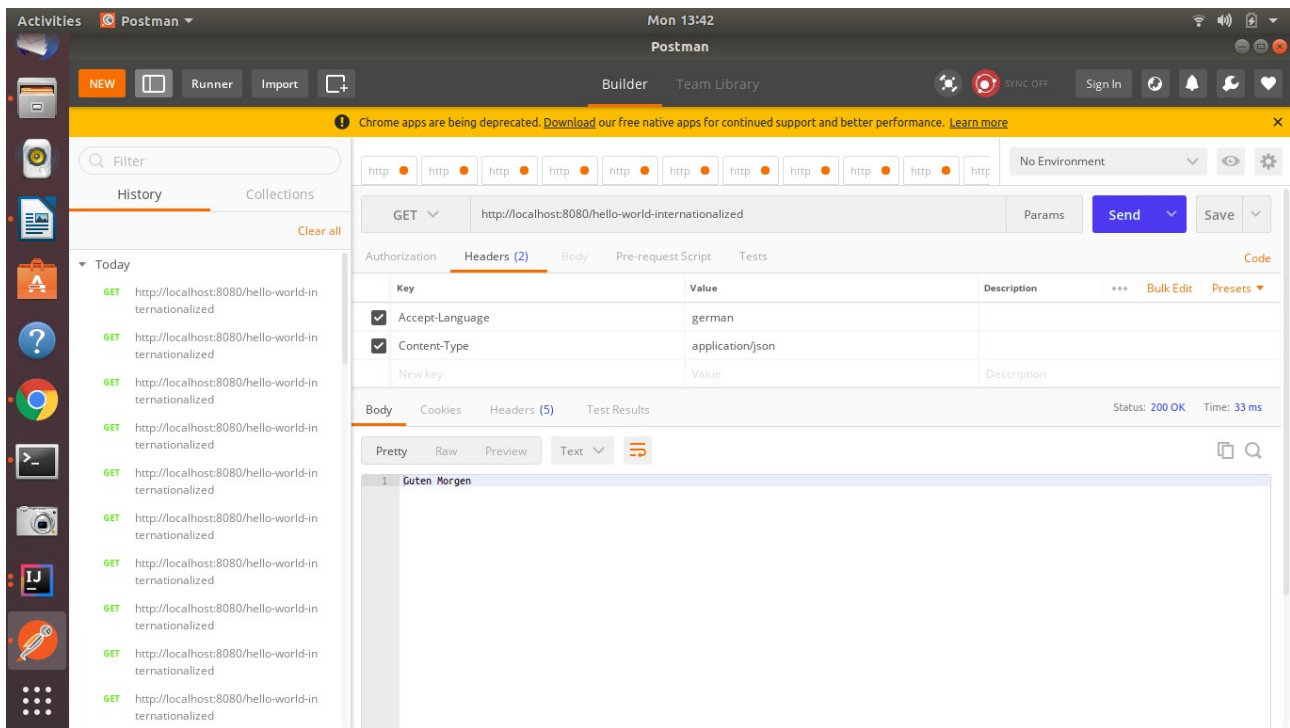
```
good.morning.message=God morgon
```





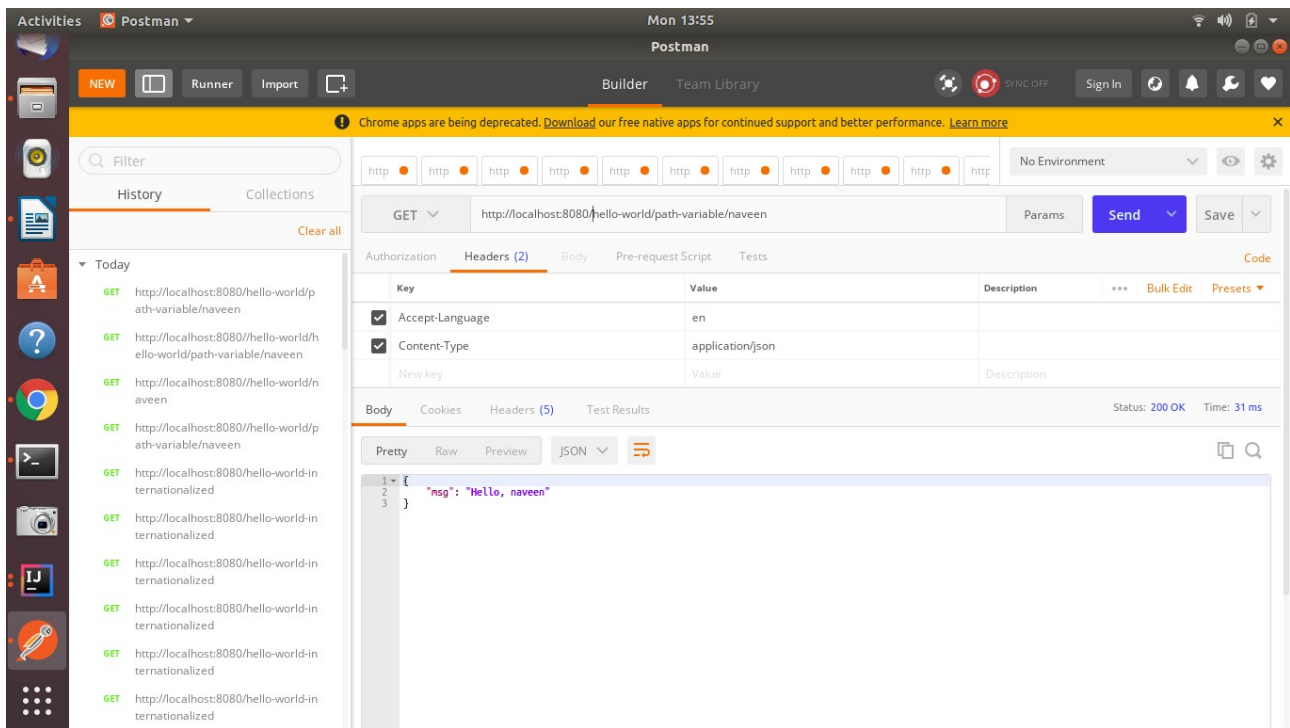
EmployeeResource.java × pom.xml (spring-boot-ex

```
1 good.morning.message=Guten Morgen
```



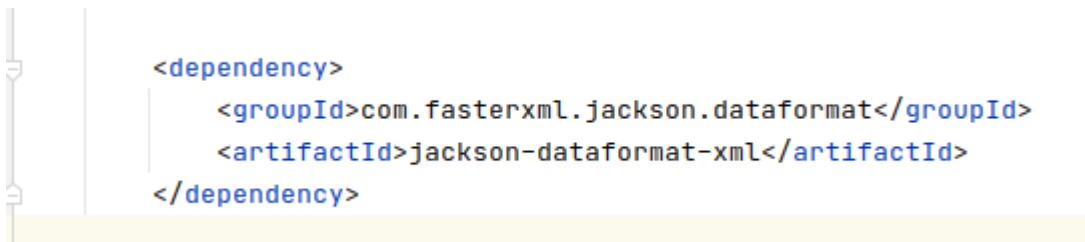
2. Create a GET request which takes "username" as param and shows a localized message "Hello Username". (Use parameters in message properties)

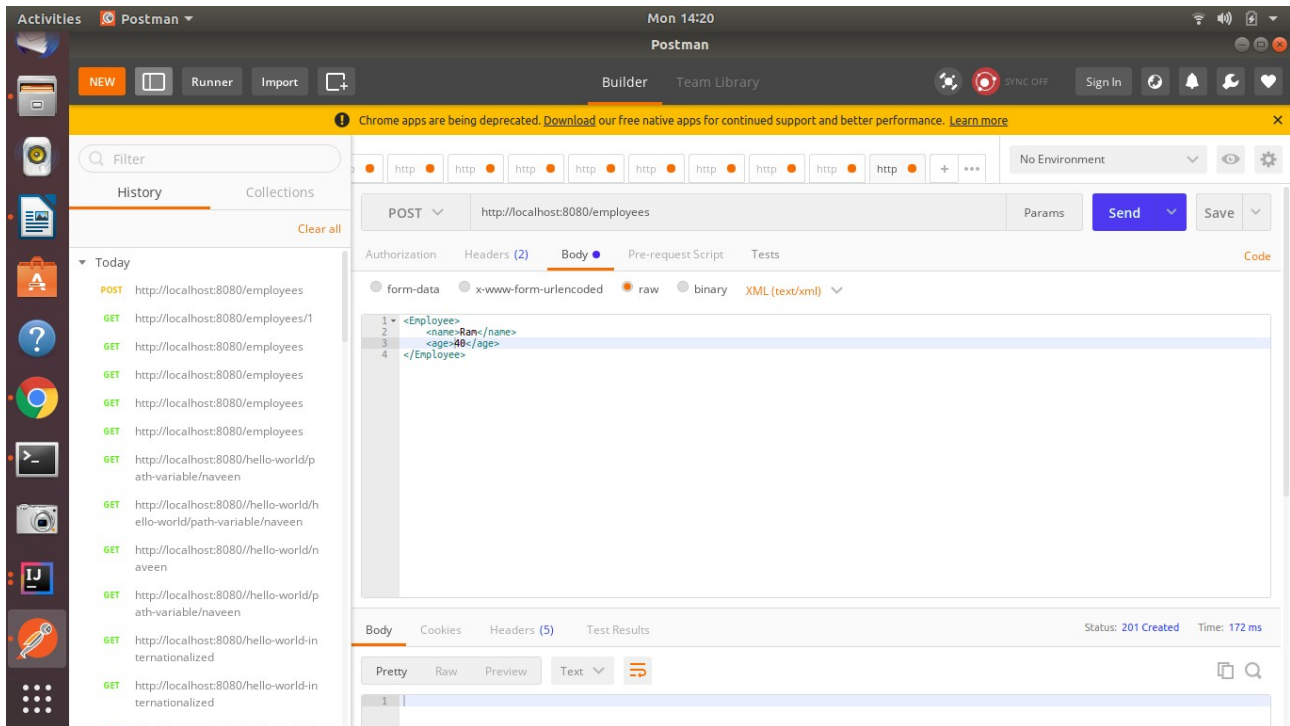
```
///hello-world/path-variable/Naveen
@GetMapping(path = "/hello-world/path-variable/{name}")
public HelloWorldBean helloWorldPathVariable(@PathVariable String name) {
    return new HelloWorldBean(String.format("Hello, %s", name));
}
```



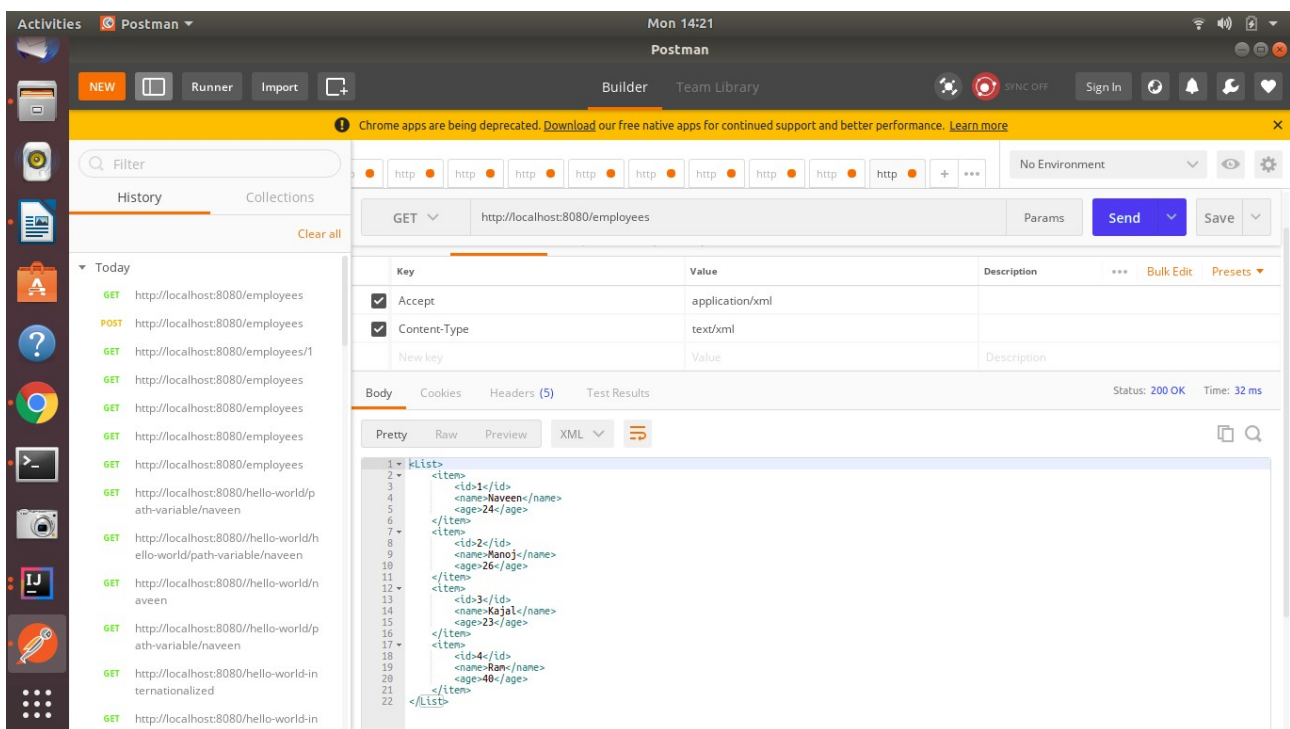
*Content Negotiation

3. Create POST Method to create user details which can accept XML for user creation.





4. Create GET Method to fetch the list of users in XML format.



*Swagger

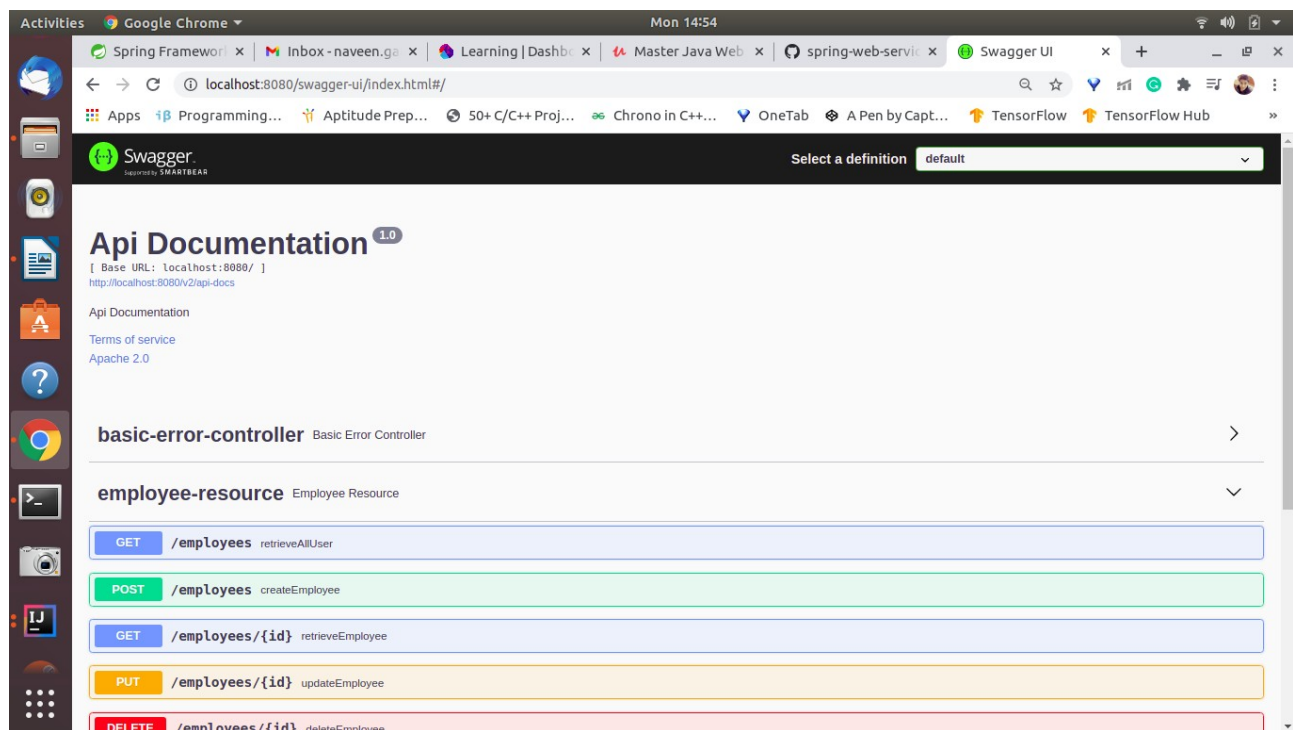
5. Configure swagger plugin and create document of following methods:

Get details of User using GET request.

Save details of the user using POST request.

Delete a user using DELETE request.

```
<dependency>  
  <groupId>io.springfox</groupId>  
  <artifactId>springfox-boot-starter</artifactId>  
  <version>3.0.0</version>  
</dependency>
```



Activities Google Chrome Mon 14:55

Spring Framework x Inbox - naveen.ga x Learning | Dashbo x Master Java Web x spring-web-servi x Swagger UI x

localhost:8080/swagger-ui/index.html#/employee-resource/retrieveAllUserUsingGET

employee-resource Employee Resource

GET /employees retrieveAllUser

Parameters Try it out

No parameters

Responses Response content type */*

Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

Example Value | Model

```
{  "age": 0,  "id": 0,  "name": "string"}
```

Activities Google Chrome Mon 14:56

Spring Framework x Inbox - naveen.ga x Learning | Dashbo x Master Java Web x spring-web-servi x Swagger UI x

localhost:8080/swagger-ui/index.html#/employee-resource/createEmployeeUsingPOST

employee-resource Employee Resource

POST /employees createEmployee

Parameters Try it out

Name	Description
employee <small>required</small> object (body)	employee

Example Value | Model

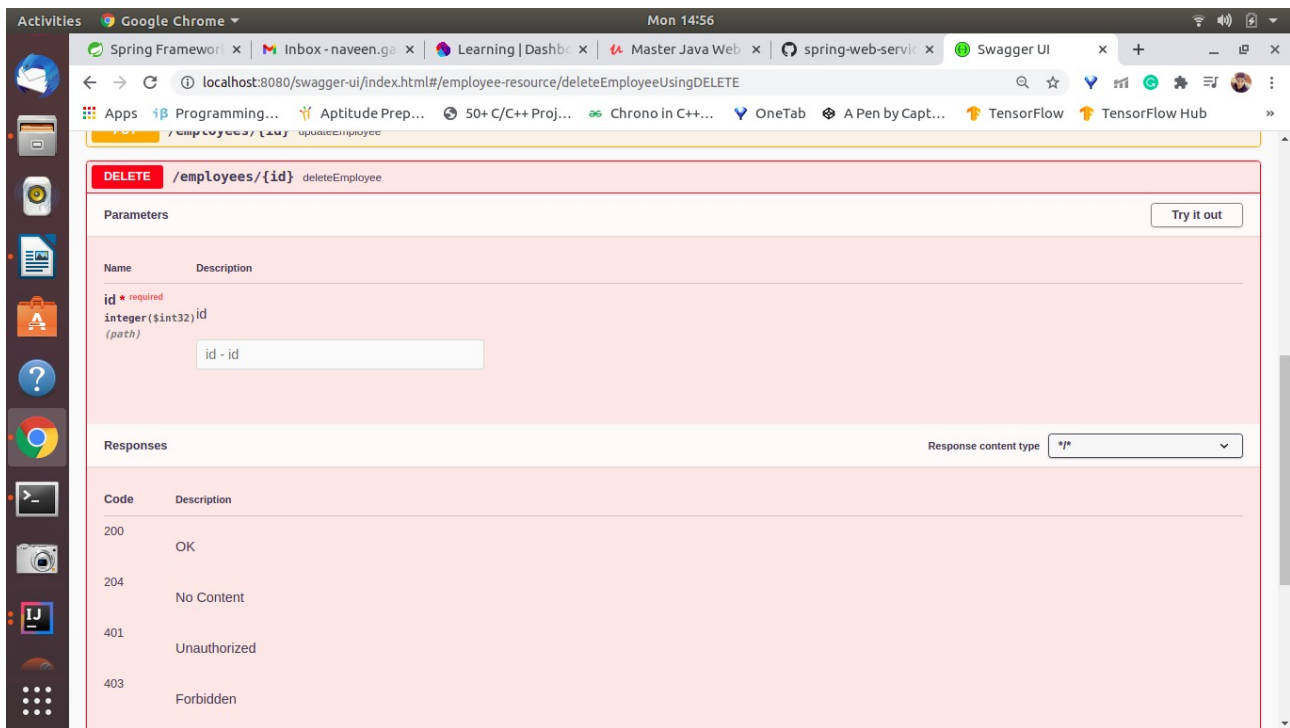
```
{  "age": 0,  "id": 0,  "name": "string"}
```

Parameter content type application/json

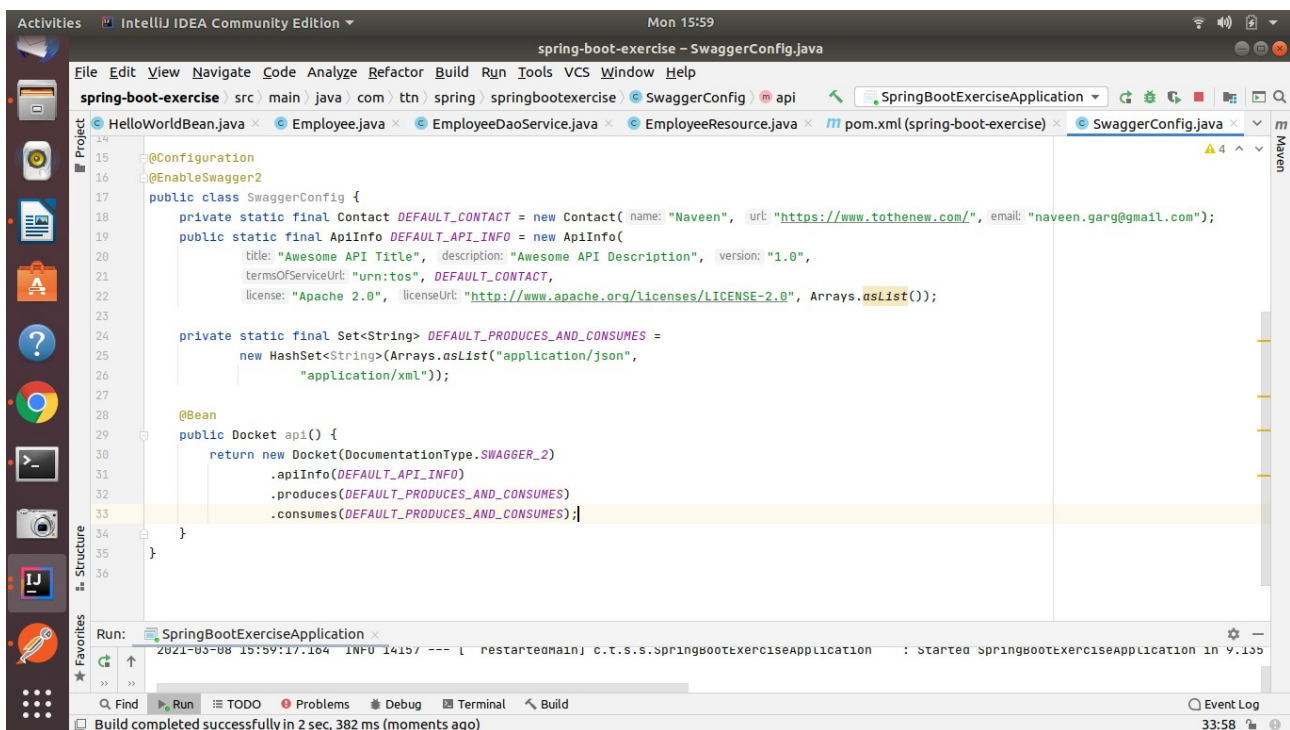
Responses Response content type */*

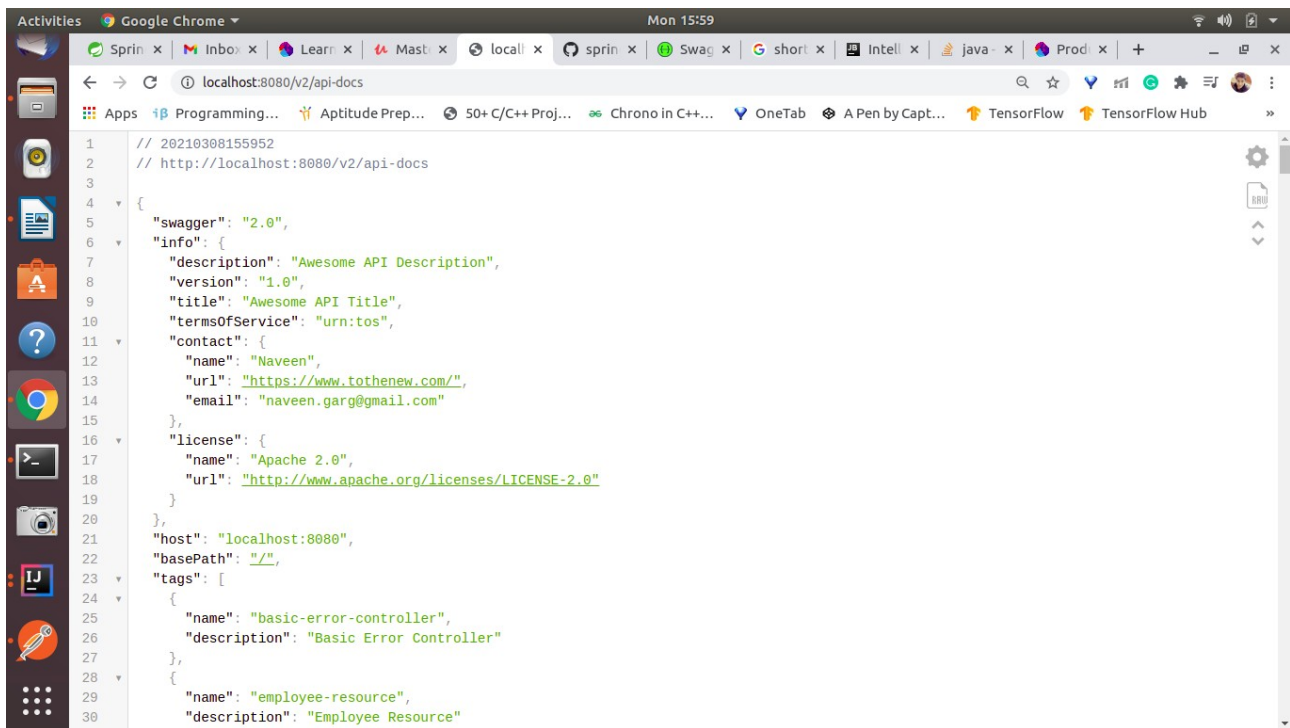
Code	Description
200	OK

Example Value | Model



7. In swagger documentation, add the description of each class and URI so that in swagger UI the purpose of class and URI is clear.



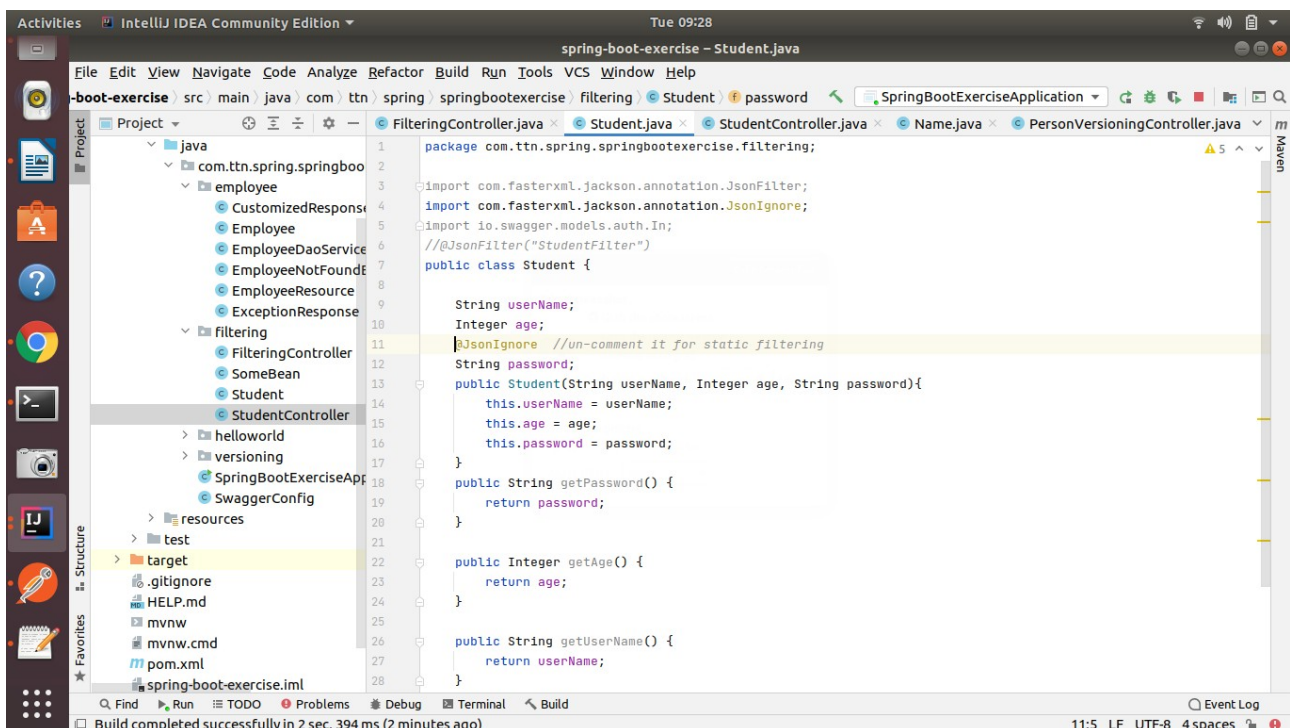


The screenshot shows a Google Chrome browser window with the address bar displaying `localhost:8080/v2/api-docs`. The page content is a JSON Swagger API definition. The JSON structure includes a `swagger` version of `2.0`, an `info` object with a description, version `1.0`, title `Awesome API Title`, terms of service `urn:tos`, and contact information for Naveen. It also includes a `license` object for Apache 2.0. The `host` is `localhost:8080` and the `basePath` is `/`. The `tags` array contains two entries: `basic-error-controller` and `employee-resource`.

```
1 // 20210308155952
2 // http://localhost:8080/v2/api-docs
3
4 {
5   "swagger": "2.0",
6   "info": {
7     "description": "Awesome API Description",
8     "version": "1.0",
9     "title": "Awesome API Title",
10    "termsOfService": "urn:tos",
11    "contact": {
12      "name": "Naveen",
13      "url": "https://www.tothenew.com/",
14      "email": "naveen.garg@gmail.com"
15    },
16    "license": {
17      "name": "Apache 2.0",
18      "url": "http://www.apache.org/licenses/LICENSE-2.0"
19    }
20  },
21  "host": "localhost:8080",
22  "basePath": "/",
23  "tags": [
24    {
25      "name": "basic-error-controller",
26      "description": "Basic Error Controller"
27    },
28    {
29      "name": "employee-resource",
30      "description": "Employee Resource"
31    }
32  ]
33 }
```

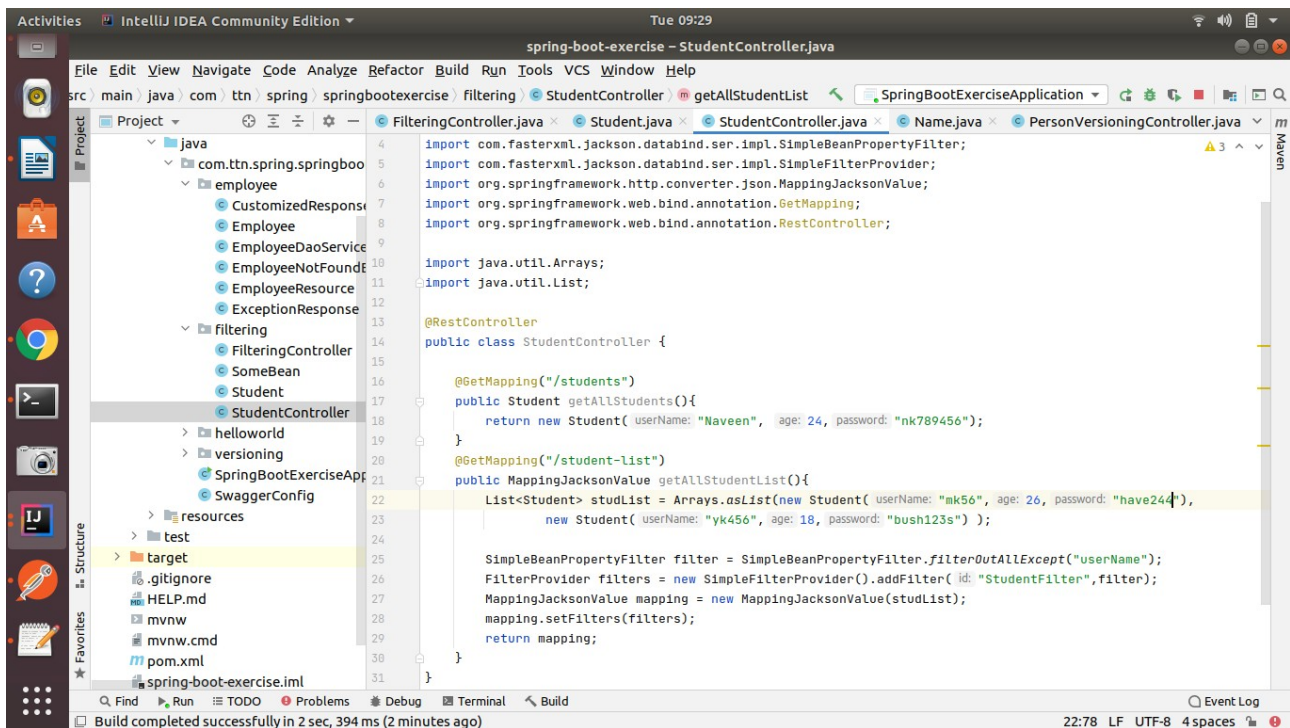
*Static and Dynamic filtering

8. Create API which saves details of User (along with the password) but on successfully saving returns only non-critical data. (Use static filtering)

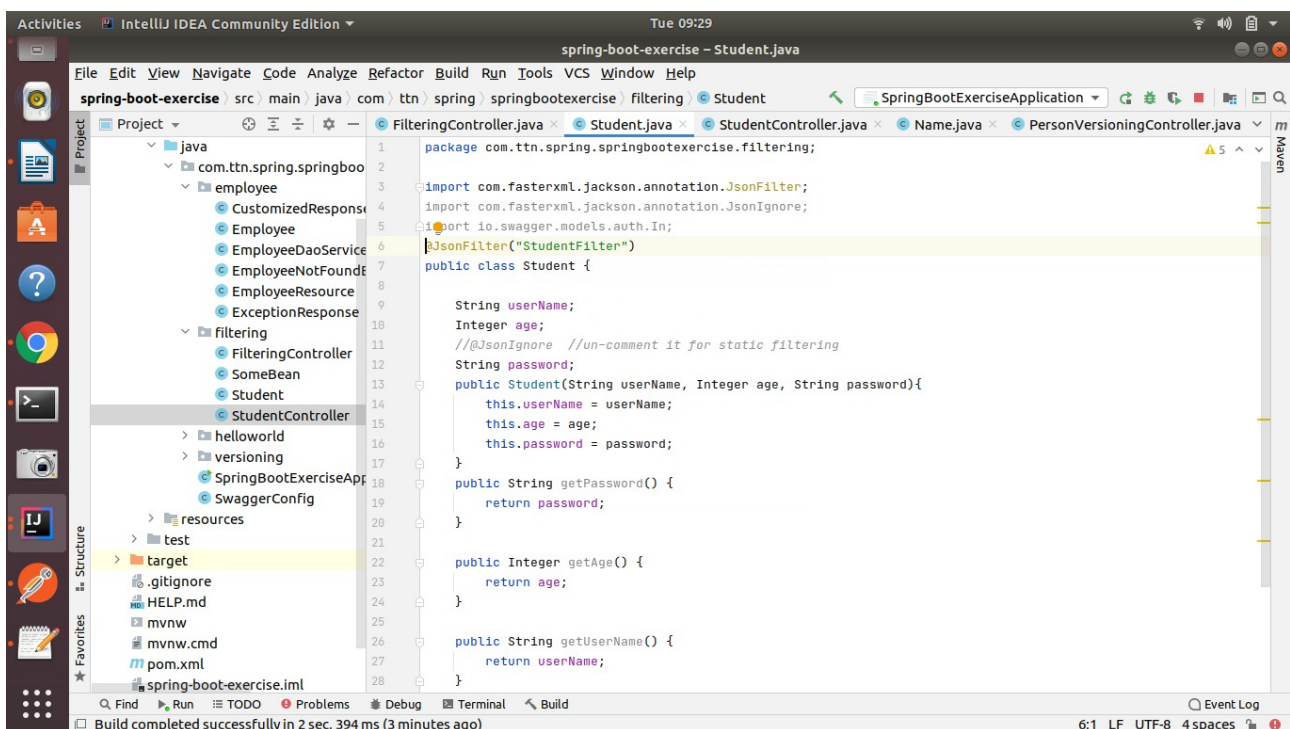


The screenshot shows the IntelliJ IDEA Community Edition interface. The project is `spring-boot-exercise`. The `src/main/java/com/ttn/spring/springbootexercise/filtering` package is selected. The `Student.java` file is open, showing the `Student` class. The class has fields `userName`, `age`, and `password`. The `password` field is annotated with `@JsonIgnore` and a comment `//un-comment it for static filtering`. The `Student` class has a constructor and three getters: `getPassword()`, `getAge()`, and `getUserName()`.

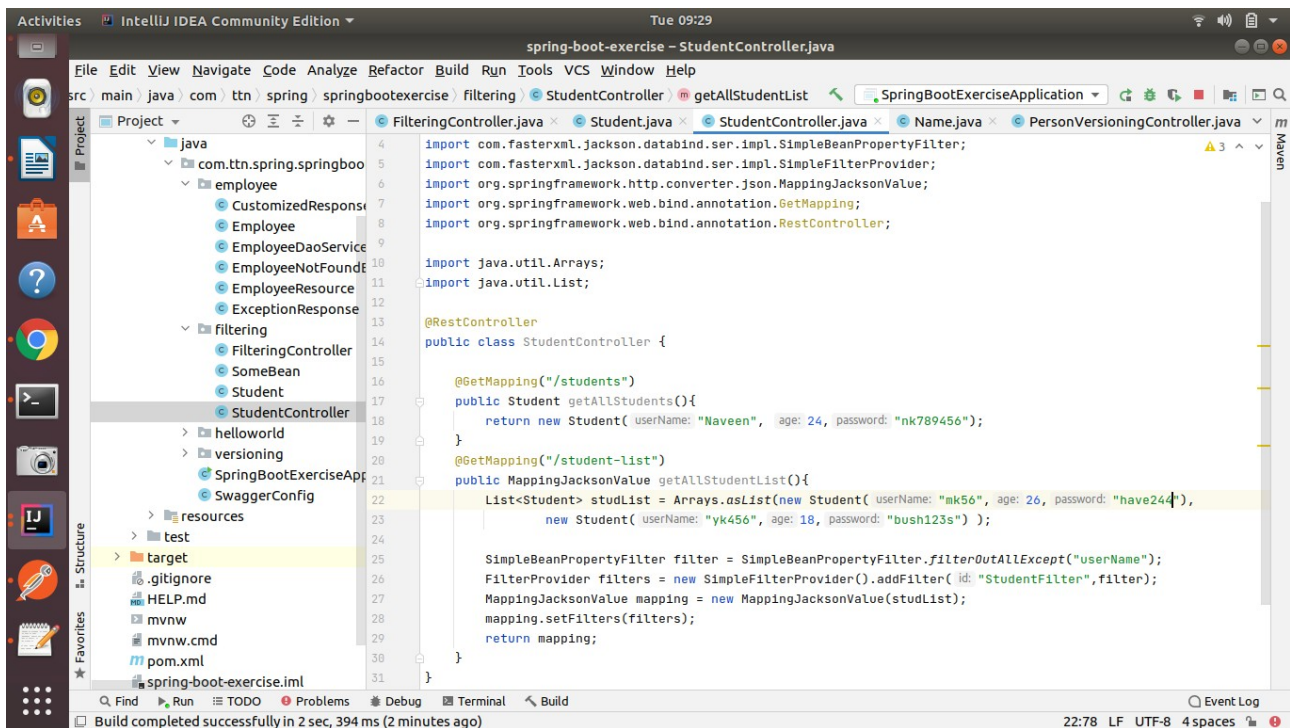
```
1 package com.ttn.spring.springbootexercise.filtering;
2
3 import com.fasterxml.jackson.annotation.JsonIgnore;
4 import com.fasterxml.jackson.annotation.JsonIgnore;
5 import io.swagger.models.auth.In;
6 // @JsonIgnore("StudentFilter")
7 public class Student {
8
9     String userName;
10    Integer age;
11    @JsonIgnore //un-comment it for static filtering
12    String password;
13
14    public Student(String userName, Integer age, String password){
15        this.userName = userName;
16        this.age = age;
17        this.password = password;
18    }
19
20    public String getPassword() {
21        return password;
22    }
23
24    public Integer getAge() {
25        return age;
26    }
27
28    public String getUserName() {
29        return userName;
30    }
31 }
```



9. Create another API that does the same by using Dynamic Filtering.



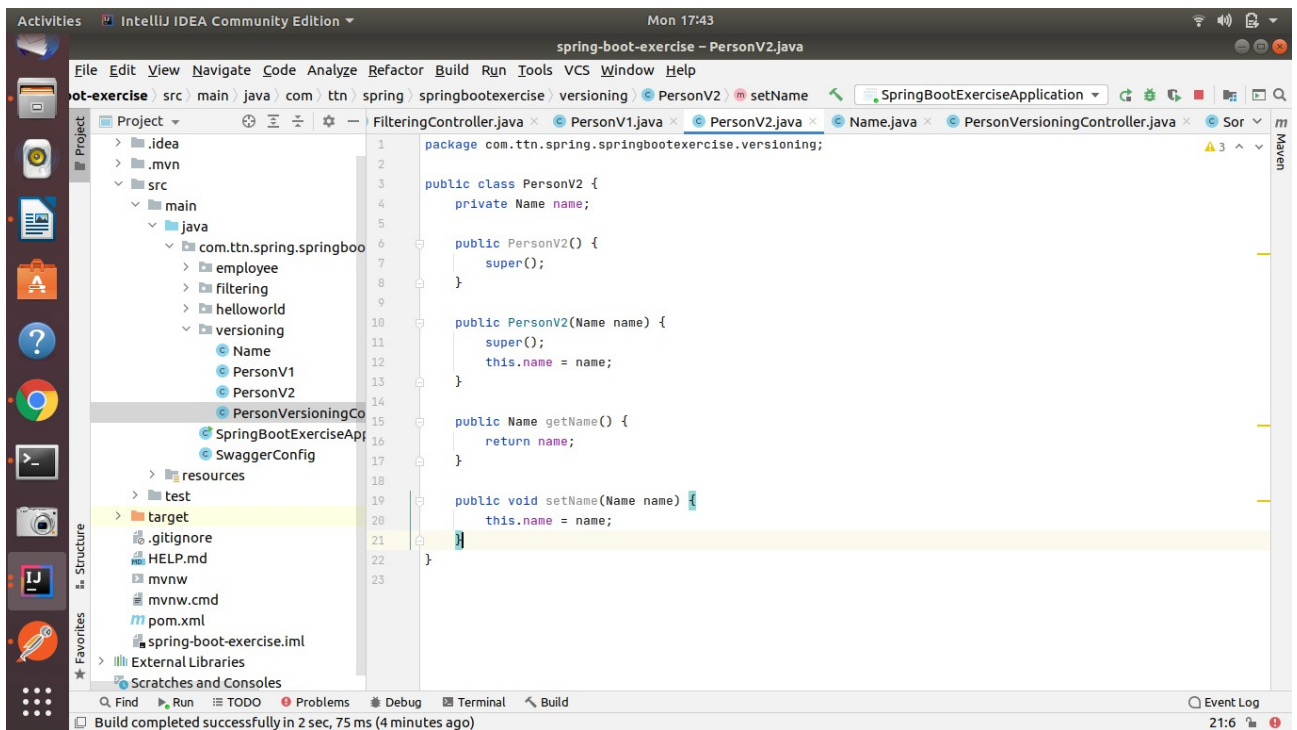
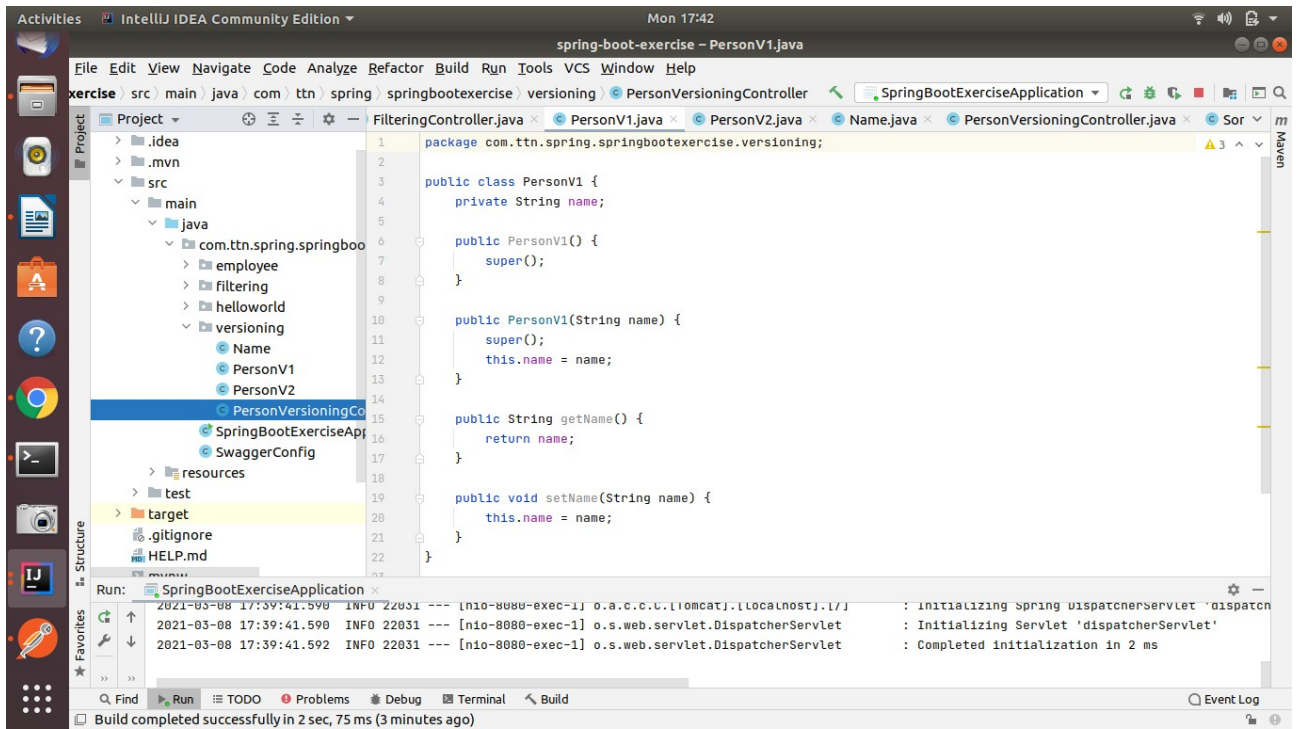
*Versioning Restful APIs

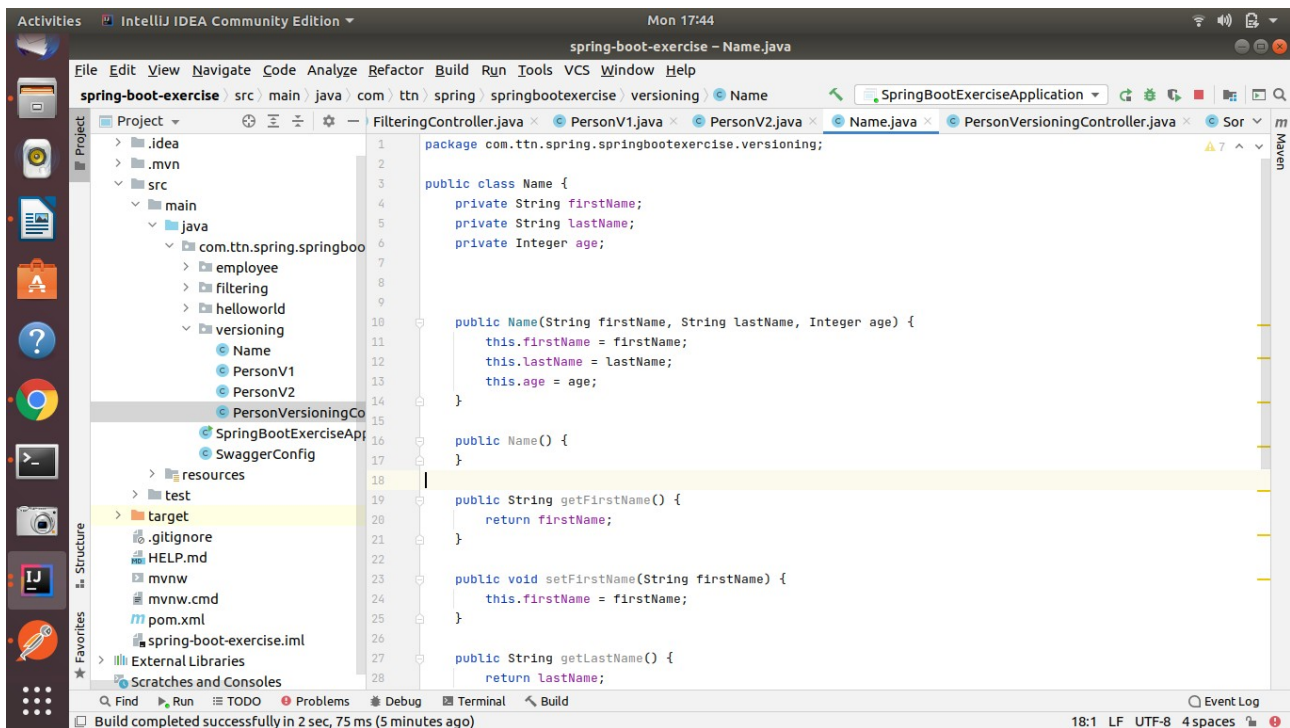


10. Create 2 API for showing user details. The first api should return only basic details of the user and the other API should return more/enhanced details of the user,

Now apply versioning using the following methods:

- **MimeType Versioning**
- **Request Parameter versioning**
- **URI versioning**
- **Custom Header Versioning**





```

package com.ttn.spring.springbootexercise.versioning;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class PersonVersioningController {
    //URI Versioning
    @GetMapping("v1/person")
    public PersonV1 personV1() {
        return new PersonV1("Naveen");
    }
    @GetMapping("v2/person")
    public PersonV2 personV2() {
        return new PersonV2(new Name("Naveen", "Aggarwal", 24));
    }
    //Request Parameter versioning
    @GetMapping(value = "/person/param", params = "version=1")
    public PersonV1 paramV1() {
        return new PersonV1("Naveen");
    }
    @GetMapping(value = "/person/param", params = "version=2")
    public PersonV2 paramV2() {
        return new PersonV2(new Name("Naveen", "Aggarwal", 24));
    }
    //Custom Header Versioning
    @GetMapping(value = "/person/header", headers = "API-VERSION=1")
    public PersonV1 headerV1() {
        return new PersonV1("Naveen");
    }
    @GetMapping(value = "/person/header", headers = "API-VERSION=2")
    public PersonV2 headerV2() {
        return new PersonV2(new Name("Naveen", "Aggarwal", 24));
    }
    //MimeType Versioning
    @GetMapping(value = "/person/produces", produces = "application/vnd.company.app-
v1+json")
    public PersonV1 producesV1() {

```

```

        return new PersonV1("Naveen");
    }
    @GetMapping(value = "/person/produces", produces = "application/vnd.company.app-
v2+json")
    public PersonV2 producesV2() {
        return new PersonV2(new Name("Naveen", "Aggarwal",24));
    }
}

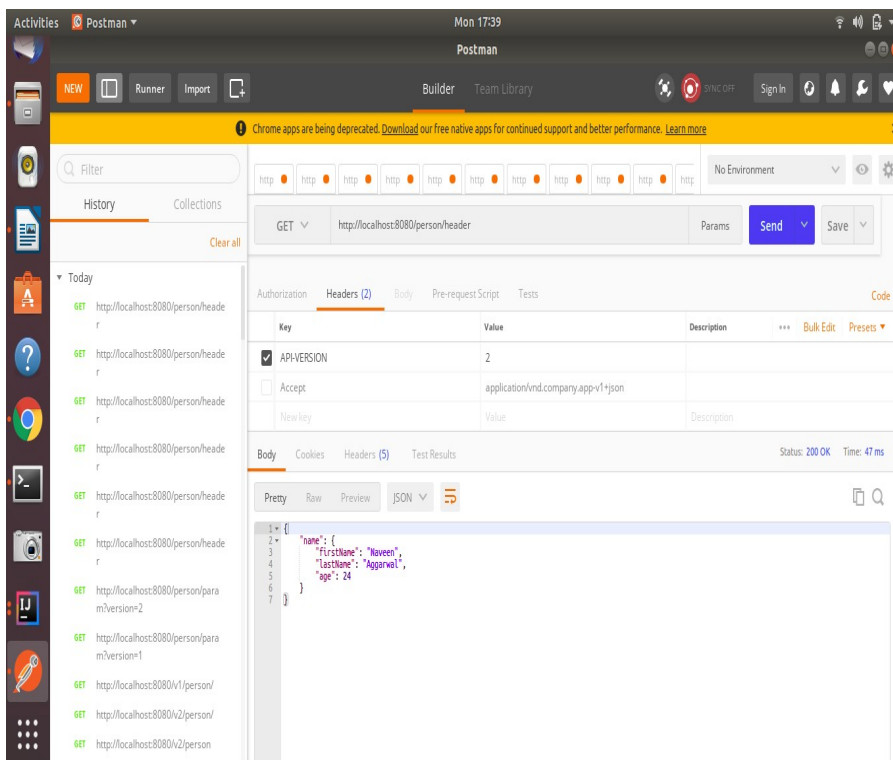
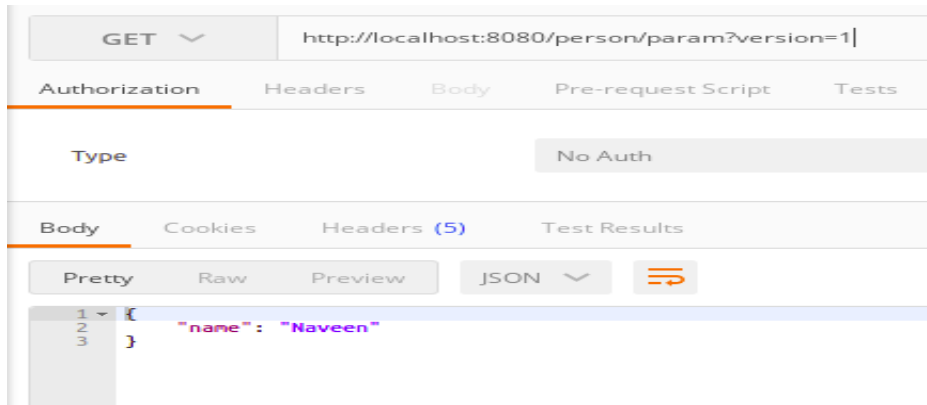
```

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

- Method:** GET
- URL:** http://localhost:8080/v1/person/
- Authorization:** No Auth
- Body:** { "name": "Naveen" }

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

- Method:** GET
- URL:** http://localhost:8080/v2/person/
- Authorization:** No Auth
- Body:** { "name": { "firstName": "Naveen", "lastName": "Aggarwal", "age": 24 } }



11. Configure hateoas with your springboot application. Create an api which returns User Details along with url to show all topics.

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


Activities IntelliJ IDEA Community Edition Mon 12:35

spring-boot-exercise - EmployeeResource.java

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

src \ main \ java \ com \ ttn \ spring \ springbootexercise \ employee \ EmployeeResource \ retrieveUser SpringBootExerciseApplication

Project Employee.java EmployeeDaoService.java EmployeeResource.java pom.xml (spring-boot-exercise) EmployeeNotFoundException.java ExceptionRe

```
39
40
41 @GetMapping("/employees/{id}")
42 public EntityModel<Employee> retrieveUser(@PathVariable int id) {
43     Employee emp = service.findOne(id);
44
45     if(emp==null)
46         throw new EmployeeNotFoundException("id-"+ id);
47
48     EntityModel<Employee> resource = EntityModel.of(emp);
49
50     WebMvcLinkBuilder linkTo =
51         linkTo(methodOn(this.getClass()).retrieveAllUser());
52
53     resource.add(linkTo.withRel("all-users"));
54
55     //HATEOAS
56     return resource;
57
58
59
60
```

Run: SpringBootExerciseApplication

```
2021-03-08 12:33:31.135 INFO 10527 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatch
2021-03-08 12:33:31.136 INFO 10527 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-03-08 12:33:31.137 INFO 10527 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
```

Q Find Run TODO Problems Terminal Build Event Log

Build completed successfully in 2 sec, 362 ms (2 minutes ago) 47:9 LF UTF-8 4 spaces

← → ↻ ⓘ localhost:8080/employees/1

📱 Apps ⓘ Programming... 🏆 Aptitude Prep... 🌐 50+ C/C++ Proj... ⌵

```
1 // 20210308123605
2 // http://localhost:8080/employees/1
3
4 {
5   "id": 1,
6   "name": "Naveen",
7   "age": 24,
8   "_links": {
9     "all-users": {
10      "href": "http://localhost:8080/employees"
11    }
12  }
13 }
```

Activities Google Chrome Mon 17:47

Spring x Inbox x Learni x Master x spring x localh x Swagg x localh x localh x Spring x +

localhost:8080/v2/api-docs

Apps iB Programming... Aptitude Prep... 50+ C/C++ Proj... Chrono in C++... OneTab A Pen by Capt... TensorFlow TensorFlow Hub

```
1 // 20210308155952
2 // http://localhost:8080/v2/api-docs
3
4 {
5   "swagger": "2.0",
6   "info": {
7     "description": "Awesome API Description",
8     "version": "1.0",
9     "title": "Awesome API Title",
10    "termsOfService": "urn:tos",
11    "contact": {
12      "name": "Naveen",
13      "url": "https://www.tothenew.com/",
14      "email": "naveen.garg@gmail.com"
15    },
16    "license": {
17      "name": "Apache 2.0",
18      "url": "http://www.apache.org/licenses/LICENSE-2.0"
19    }
20  },
21  "host": "localhost:8080",
22  "basePath": "/",
23  "tags": [
24    {
25      "name": "basic-error-controller",
26      "description": "Basic Error Controller"
27    },
28    {
29      "name": "employee-resource",
30      "description": "Employee Resource"
```