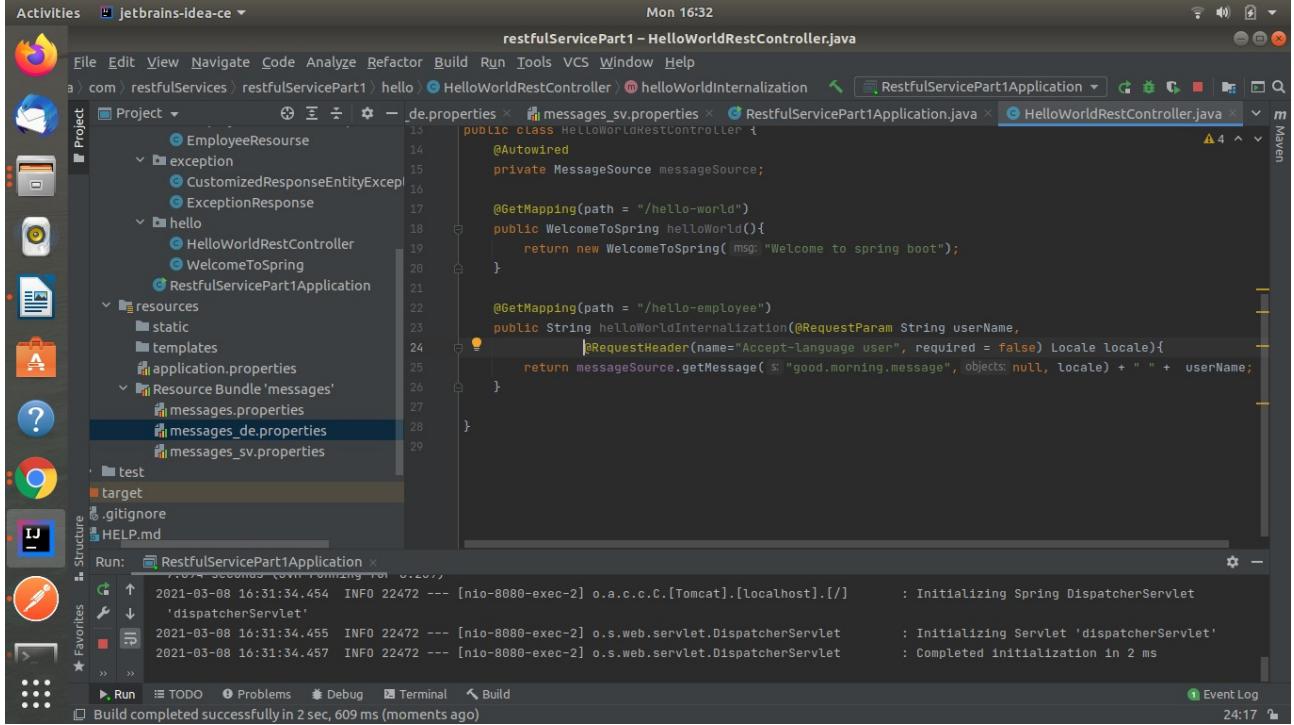


## \*Internationalization

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

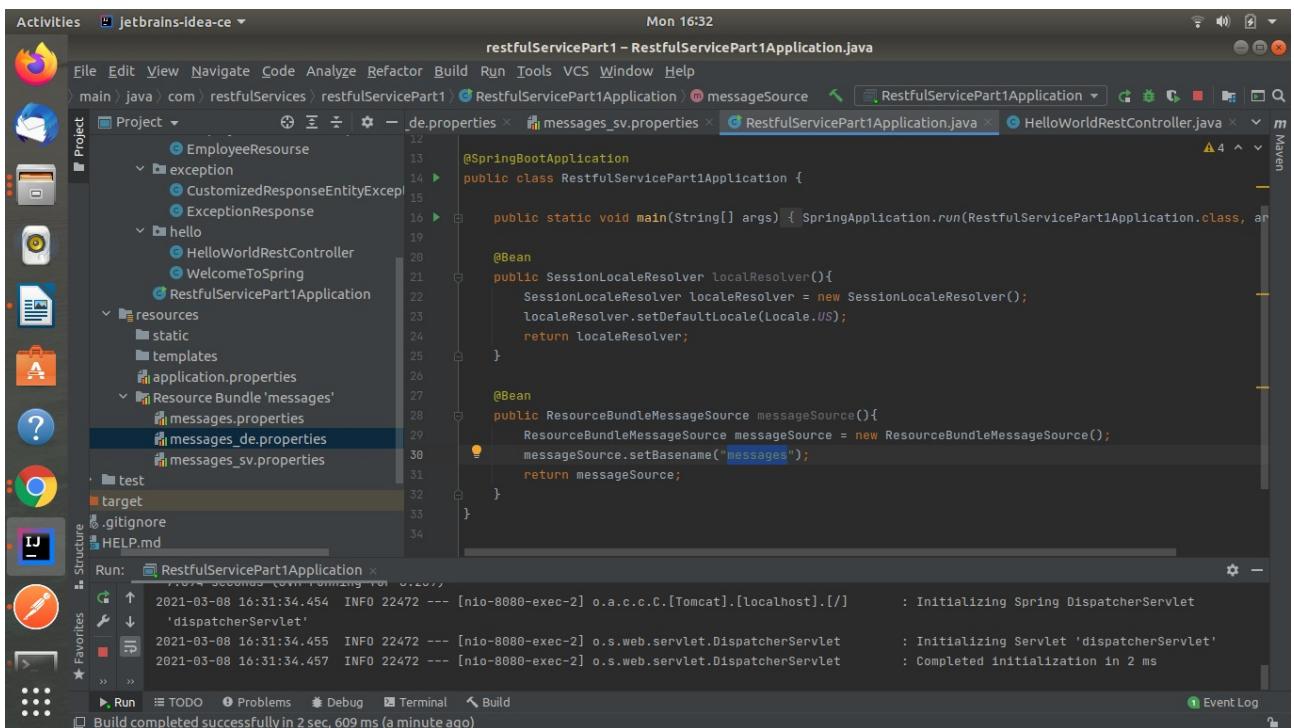


```
restfulServicePart1 - HelloWorldRestController.java
Mon 16:32
public class HelloWorldRestController {
    @Autowired
    private MessageSource messageSource;

    @GetMapping(path = "/hello-world")
    public WelcomeToSpring helloWorld(){
        return new WelcomeToSpring( msg: "Welcome to spring boot");
    }

    @GetMapping(path = "/hello-employee")
    public String helloWorldInternalization(@RequestParam String userName,
                                            @RequestHeader(name="Accept-Language" user, required = false) Locale locale){
        return messageSource.getMessage( "good.morning.message", (Objects: null, locale) + " " + userName;
    }
}
```

The screenshot shows the IntelliJ IDEA interface with the code editor open to `HelloWorldRestController.java`. The code defines two `@GetMapping` methods. The first returns a `WelcomeToSpring` object with a fixed message. The second method uses `MessageSource` to retrieve a message from a properties file based on the user's locale (determined by the `Accept-Language` header). The project structure on the left shows resources for English, German, and Swedish.



```
restfulServicePart1 - RestfulServicePart1Application.java
Mon 16:32
@SpringBootApplication
public class RestfulServicePart1Application {
    public static void main(String[] args) { SpringApplication.run(RestfulServicePart1Application.class, args); }

    @Bean
    public SessionLocaleResolver localeResolver(){
        SessionLocaleResolver localeResolver = new SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.US);
        return localeResolver;
    }

    @Bean
    public ResourceBundleMessageSource messageSource(){
        ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
        messageSource.setBasename("messages");
        return messageSource;
    }
}
```

The screenshot shows the IntelliJ IDEA interface with the code editor open to `RestfulServicePart1Application.java`. This is a Spring Boot configuration class. It uses `@SpringBootApplication` to enable auto-configuration. It defines a `main` method for starting the application. It also contains two `@Bean` definitions: one for a `SessionLocaleResolver` which sets the default locale to US, and another for a `ResourceBundleMessageSource` which reads messages from a `messages` properties file.

The screenshot shows the Postman application interface. The left sidebar is titled "My Workspace" and contains sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace shows a single API endpoint: `http://localhost:8080/employees/1`. The request method is set to GET. The Headers tab is selected, showing a header entry for X-API-VERSION. The Body tab displays a JSON response with fields id, name, age, and links. The response status is 200 OK with a duration of 169 ms and a size of 269 B.

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

The screenshot shows the Postman application interface. The left sidebar is titled "My Workspace" and contains sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace shows a single API endpoint: `http://localhost:8080/hello-employee?userName=Nitin`. The request method is set to GET. The Params tab is selected, showing a query parameter entry for userName with the value Nitin. The response status is 200 OK with a duration of 290 ms and a size of 175 B, displaying the localized message "Hello Nitin".

The screenshot shows the Postman application interface. In the top navigation bar, it says "Mon 16:31" and "Postman". The left sidebar has sections for "Collections", "APIs", "Environments", "Mock Servers", "Monitors", and "History". The main workspace is titled "My Workspace" with tabs for "New" and "Import". A search bar at the top right says "Search Postman". Below the search bar, there are three requests listed: "GET http://localhost:8080/hello-employee?userName=Nitin" (status 200 OK), "GET http://localhost:8080/hello-employee?userName=Nitin" (status 200 OK), and "GET http://localhost:8080/hello-employee?userName=Nitin" (status 200 OK). The middle request's details are shown: method "GET", URL "http://localhost:8080/hello-employee?userName=Nitin", Headers tab selected, showing "Accept-language: de", Body tab selected, showing "Value: Hello Nitin", and a status bar at the bottom right showing "200 OK 290 ms 175 B".

The screenshot shows the IntelliJ IDEA interface. The title bar says "Activities" and "Tue 12:07". The central area shows the code editor with the file "HelloWorldRestController.java" open. The code is as follows:

```
restfulServicePart1 - HelloWorldRestController.java
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.util.Locale;

@RestController
public class HelloWorldRestController {
    @Autowired
    private MessageSource messageSource;

    @GetMapping(path = "/hello-world")
    public WelcomeToSpring helloWorld() { return new WelcomeToSpring( msg: "Welcome to spring boot"); }

    @GetMapping(path = "/hello-employee")
    public String helloWorldInternalization(@RequestParam String userName){
        return messageSource.getMessage("good.morning.message", null,
                LocaleContextHolder.getLocale()) + " " + userName;
    }
}
```

The left sidebar shows the project structure with packages like "com.restfulServices.restfulServicePart1.hello", "Employee", "EmployeeResource", "exception", "filtering", "hello", "versioning", and "resources". The "resources" folder contains "static", "templates", and "application.properties". The "application.properties" file includes the line "Resource Bundle 'messages' messages.properties". The bottom status bar shows "Build completed successfully in 2 sec, 453 ms (today 10:39 AM)" and the time "26:49".

## \*Content Negotiation

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

The screenshot shows the Postman application interface. In the center, there is a request card for a POST method to the URL `http://localhost:8080/employees`. The 'Body' tab is selected, showing the XML payload:

```
<item>
<name>Abhishek</name>
<age>22</age>
</item>
```

The response status is 201 Created, with a time of 240 ms and a size of 173 B.

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

The screenshot shows the Postman application interface. In the center, there is a request card for a GET method to the URL `http://localhost:8080/employees`. The 'Headers' tab is selected, showing the header configuration:

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> Accept	application/xml			
Key	Value	Description		

The response status is 200 OK, with a time of 438 ms and a size of 351 B.

## \*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.

The screenshot shows the Google Chrome browser window with the address bar set to `localhost:8080/swagger-ui/index.html#/employee-resource/getEmpUsingGET`. The main content area displays a Swagger API documentation page for a `GET /employees/{id}` endpoint named `getEmp`. The `Parameters` section shows a required parameter `id` of type `integer($int32)` with a value of `2` entered into the input field. Below the parameters are sections for `Responses`, `Curl` (containing the command `curl -X GET "http://localhost:8080/employees/2" -H "accept: */*`), `Request URL` (`http://localhost:8080/employees/2`), and `Server response`.

This screenshot shows the detailed server response for the GET request made in the previous screenshot. The `Curl` section contains the same command. The `Request URL` is also shown as `http://localhost:8080/employees/2`. The `Server response` section is expanded, showing the `Code` (200) and `Details` sections. The `Response body` shows a JSON object with fields `id`, `name`, `age`, and `links`. The `links` field contains a link to the `all-employees` collection. The `Response headers` section lists standard HTTP headers like `connection`, `content-type`, `date`, `keep-alive`, and `transfer-encoding`. A `Download` button is available for the response body.

Activities Google Chrome ▾ Mon 17:40

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

}

Response headers

```
connection: keep-alive
content-type: application/hal+json
date: Mon 08 Mar 2021 12:10:09 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description
200	OK
	Example Value   Model
	{ "_links": { "empty": true }, "age": 0, "id": 0, "name": "string" }
401	Unauthorized
403	Forbidden
404	Not Found

Download

This screenshot shows the Swagger UI interface for a REST API endpoint. The URL is localhost:8080/swagger-ui/index.html#/employee-resource/getEmpUsingGET. The response section is displayed, showing the response body, headers, and various status codes (200, 401, 403, 404) with their descriptions and example values.

Activities Google Chrome ▾ Mon 17:45

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

Name	Description
emp <small>required</small>	emp
object (body)	Edit Value   Model

```
{ "age": 20,
  "name": "Nitin" }
```

Cancel

Parameter content type

application/json

Execute Clear

This screenshot shows the Swagger UI interface for a REST API endpoint. The URL is localhost:8080/swagger-ui/index.html#/employee-resource/getAllEmpUsingGET. A modal dialog is open for a parameter named 'emp' which is marked as required. The parameter type is 'object' and it is defined as a 'body'. The input field contains a JSON object with 'age' set to 20 and 'name' set to 'Nitin'. Below the input field are buttons for 'Cancel' and 'Execute', and a dropdown menu for 'Parameter content type' currently set to 'application/json'.

Activities Google Chrome ▾ Mon 17:45

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

Curl

```
curl -X POST "http://localhost:8080/employees" -H "accept: /*" -H "Content-Type: application/json" -d "{ \"age\": 20, \"name\": \"Nitin\"}"
```

Request URL

http://localhost:8080/employees

Server response

Code Details

201 Response headers

```
connection: keep-alive  
content-length: 0  
date: Mon08 Mar 2021 12:14:51 GMT  
keep-alive: timeout=60  
location: http://localhost:8080/employees/5
```

Responses

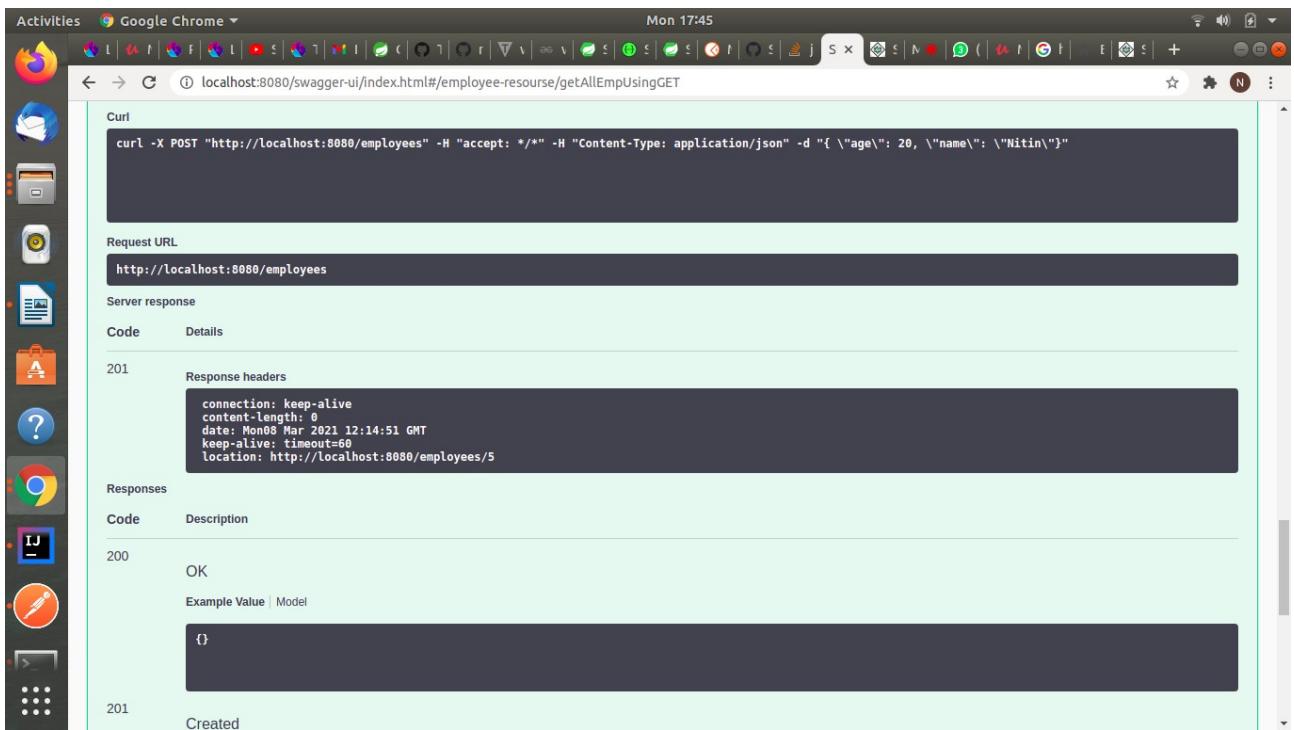
Code Description

200 OK

Example Value | Model

```
{}
```

201 Created



Activities Google Chrome ▾ Mon 17:45

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

Server response

Code Details

201 Response headers

```
connection: keep-alive  
content-length: 0  
date: Mon08 Mar 2021 12:14:51 GMT  
keep-alive: timeout=60  
location: http://localhost:8080/employees/5
```

Responses

Code Description

200 OK

Example Value | Model

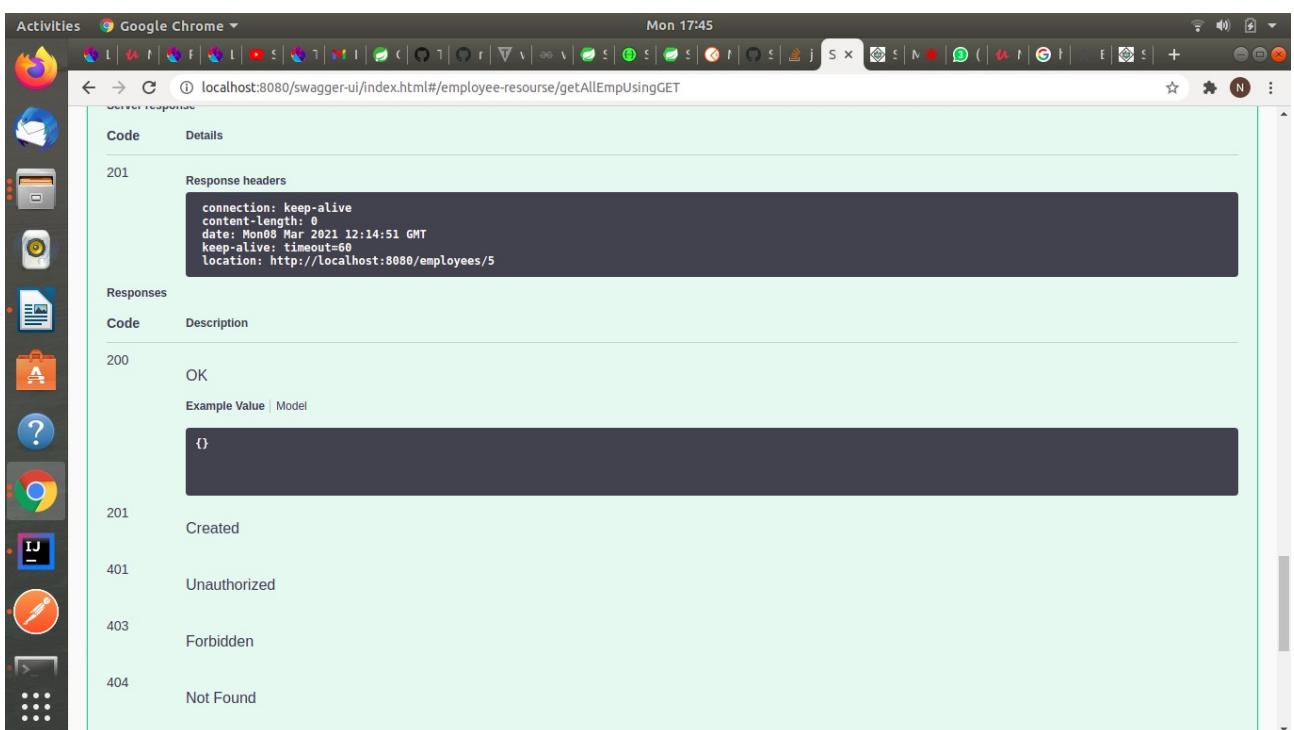
```
{}
```

201 Created

401 Unauthorized

403 Forbidden

404 Not Found



Activities Google Chrome ▾ Mon 17:46

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

**DELETE /employees/{id} deleteEmp**

Parameters

Name Description

**id** \* required  
integer(\$int32) id  
(path)

5

Execute Clear

Responses

Curl

```
curl -X DELETE "http://localhost:8080/employees/5" -H "accept: */"
```

Request URL

http://localhost:8080/employees/5

Server response

Activities Google Chrome ▾ Mon 17:46

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

Curl

```
curl -X DELETE "http://localhost:8080/employees/5" -H "accept: */"
```

Request URL

http://localhost:8080/employees/5

Server response

Code Details

200 Response headers

```
connection: keep-alive
content-length:
date: Mon 08 Mar 2021 12:16:15 GMT
keep-alive: timeout=60
```

Responses

Code Description

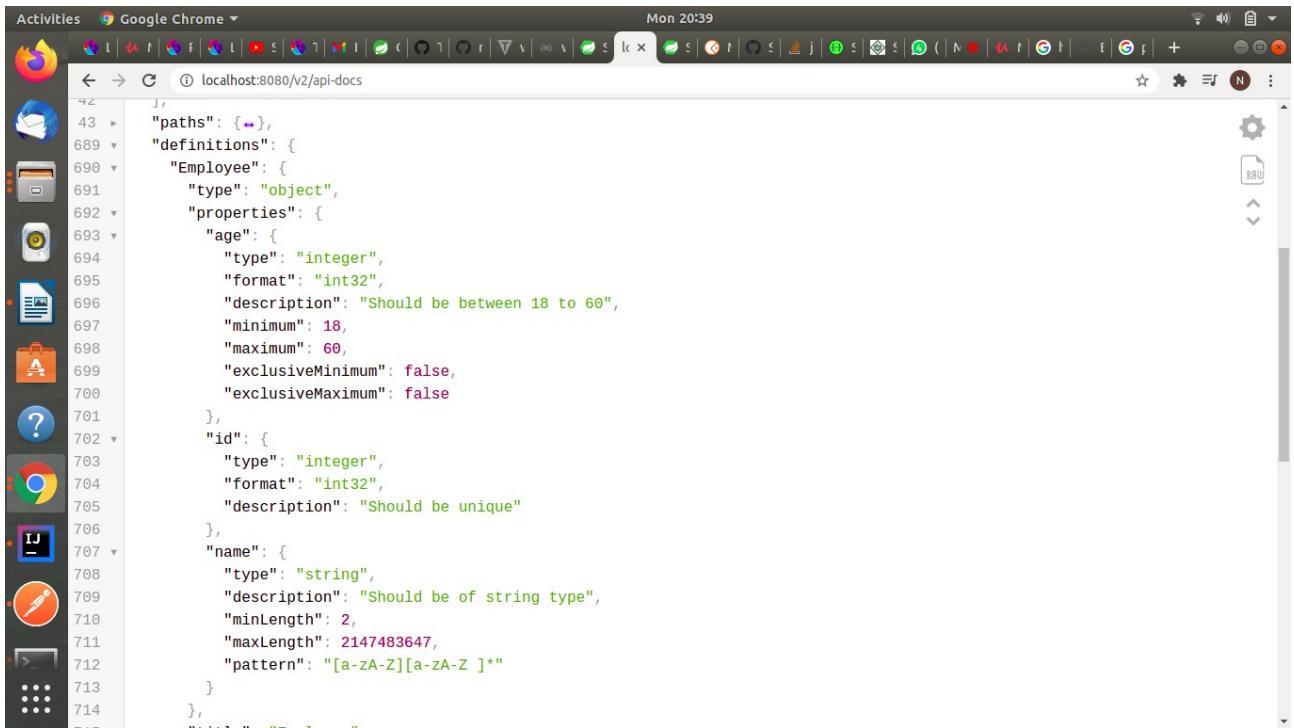
200 OK

204 No Content

401 Unauthorized

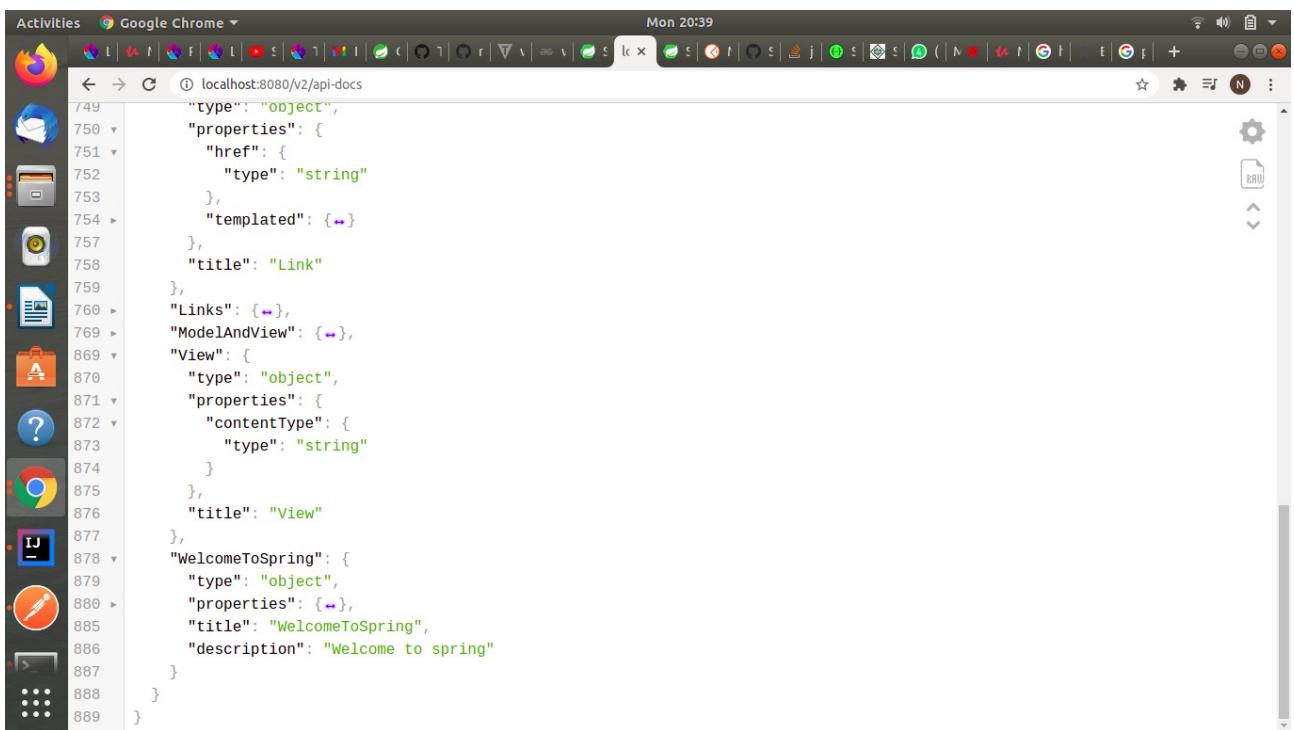
403 Forbidden

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.



```
Activities Google Chrome ▾ Mon 20:39
localhost:8080/v2/api-docs

43  "paths": { },
689  "definitions": {
690    "Employee": {
691      "type": "object",
692      "properties": {
693        "age": {
694          "type": "integer",
695          "format": "int32",
696          "description": "Should be between 18 to 60",
697          "minimum": 18,
698          "maximum": 60,
699          "exclusiveMinimum": false,
700          "exclusiveMaximum": false
701        },
702        "id": {
703          "type": "integer",
704          "format": "int32",
705          "description": "Should be unique"
706        },
707        "name": {
708          "type": "string",
709          "description": "Should be of string type",
710          "minLength": 2,
711          "maxLength": 2147483647,
712          "pattern": "[a-zA-Z][a-zA-Z ]*"
713        }
714      }
715    }
716  }
717 }
```



```
Activities Google Chrome ▾ Mon 20:39
localhost:8080/v2/api-docs

749  "type": "object",
750  "properties": {
751    "href": {
752      "type": "string"
753    },
754    "templated": { }
755  },
756  "title": "Link"
757 },
758  "Links": { },
759  "ModelAndView": { },
760  "View": {
761    "type": "object",
762    "properties": {
763      "contentType": {
764        "type": "string"
765      }
766    },
767    "title": "View"
768  },
769  "WelcomeToSpring": {
770    "type": "object",
771    "properties": { },
772    "title": "WelcomeToSpring",
773    "description": "Welcome to spring"
774  }
775 }
```

Tue 10:41

restfulServicePart1 – EmployeeResource.java

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
rt1 src main java com restfulServices restfulServicePart1 employe EmployeeResource @getEmp RestfulServicePart1Application
Project Project .idea .mvn src main java com.restfulServices.restfulServicePart1 employe EmployeeResource Employee.java StudentController.java WelcomeToSpring.java EmployeeResource.java
src main java com.restfulServices.restfulServicePart1 employe Employee EmployeeDaoService EmployeeNotFoundException EmployeeResource exception CustomizedResponseEntity ErrorResponse filtering Student StudentController hello
Employee.java EmployeeResource.java

@RestController
public class EmployeeResource {
    @Autowired
    private EmployeeDaoService service;

    @GetMapping(path = "/employees")
    @ApiModelProperty(notes = "All employees details")
    public List<Employee> getAllEmp() { return service.findAll(); }

    @GetMapping(path = "/employees/{id}")
    @ApiModelProperty(notes = "employee detail")
    public EntityModel<Employee> getEmp(@PathVariable int id){
        Employee emp = service.findOne(id);
        if(emp == null) throw new EmployeeNotFoundException("id- " + id);

        EntityModel<Employee> resource = EntityModel.of(emp);
        WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).getAllEmp());
        resource.add(linkTo.withRel("all-employees"));
        return resource;
    }
}
```

Run: RestfulServicePart1Application

```
2021-03-09 10:39:57.239 INFO 18062 --- [ restartedMain] c.r.R.RestfulServicePart1Application : Started RestfulServicePart1Application in 8.33 seconds (JVM running for 9.26)
2021-03-09 10:40:06.342 INFO 18062 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet
2021-03-09 10:40:06.342 INFO 18062 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-03-09 10:40:06.344 INFO 18062 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
```

Event Log

Build completed successfully in 2 sec, 453 ms (a minute ago)

Tue 12:10

restfulServicePart1 – Employee.java

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
rt1 src main java com restfulServices restfulServicePart1 employe Employee Employee.java HelloWorldRestController StudentController WelcomeToSpring versioning Name Person1 Person2 VersionController RestfulServicePart1Application SwaggerConfig
Project Project .idea .mvn src main java com.restfulServices.restfulServicePart1 employe Employee EmployeeDaoService EmployeeNotFoundException EmployeeResource exception CustomizedResponseEntity ErrorResponse filtering Student StudentController hello
HelloWorldRestController.java StudentController.java WelcomeToSpring.java Employee.java

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

@ApiModel(description = "Employee details")
public class Employee {
    @ApiModelProperty(notes = "Should be unique")
    Integer id;

    @ApiModelProperty(notes = "Should be of string type")
    @Pattern(regexp = "[a-zA-Z][a-zA-Z]*")
    @Size(min = 2)
    String name;

    @ApiModelProperty(notes = "Should be between 18 to 60")
    @Min(value = 18)
    @Max(value = 60)
    Integer age;

    protected Employee(){}
    Employee(Integer id, String name, Integer age){}
```

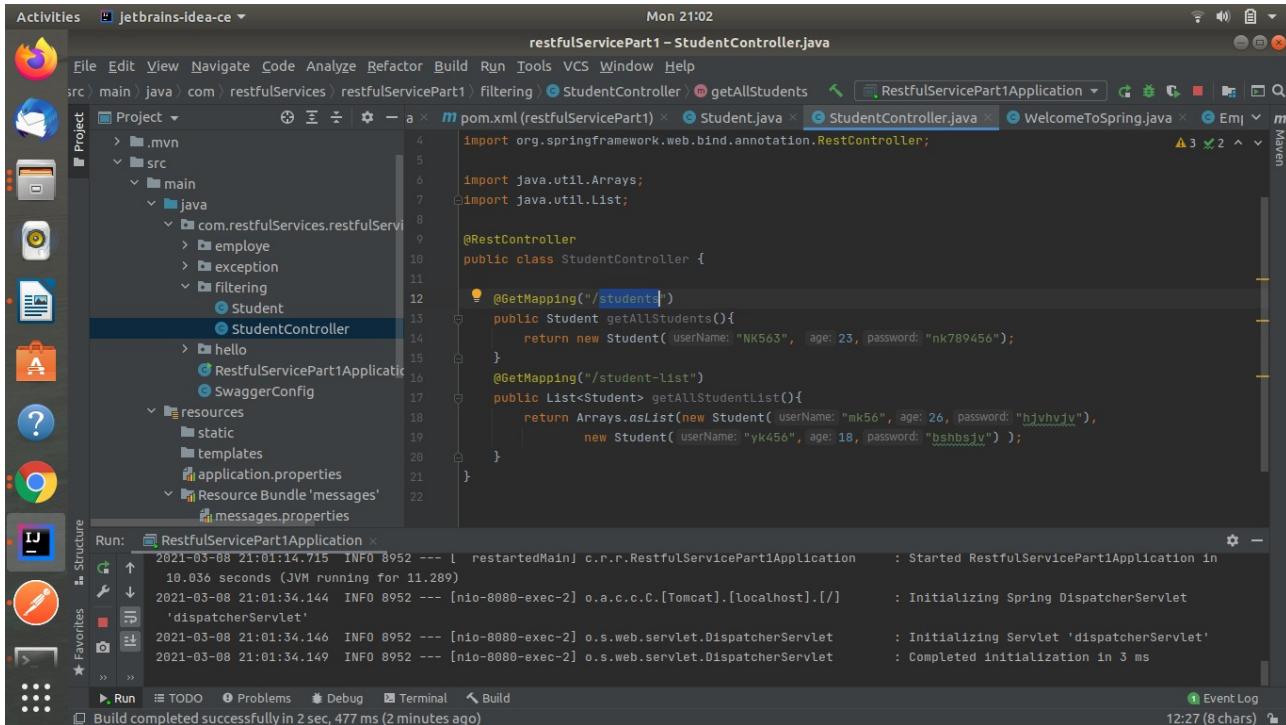
Run: RestfulServicePart1Application

Event Log

Build completed successfully in 2 sec, 453 ms (today 10:39 AM)

## \*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 interface with the code editor open to `StudentController.java`. The code defines a REST controller for students:

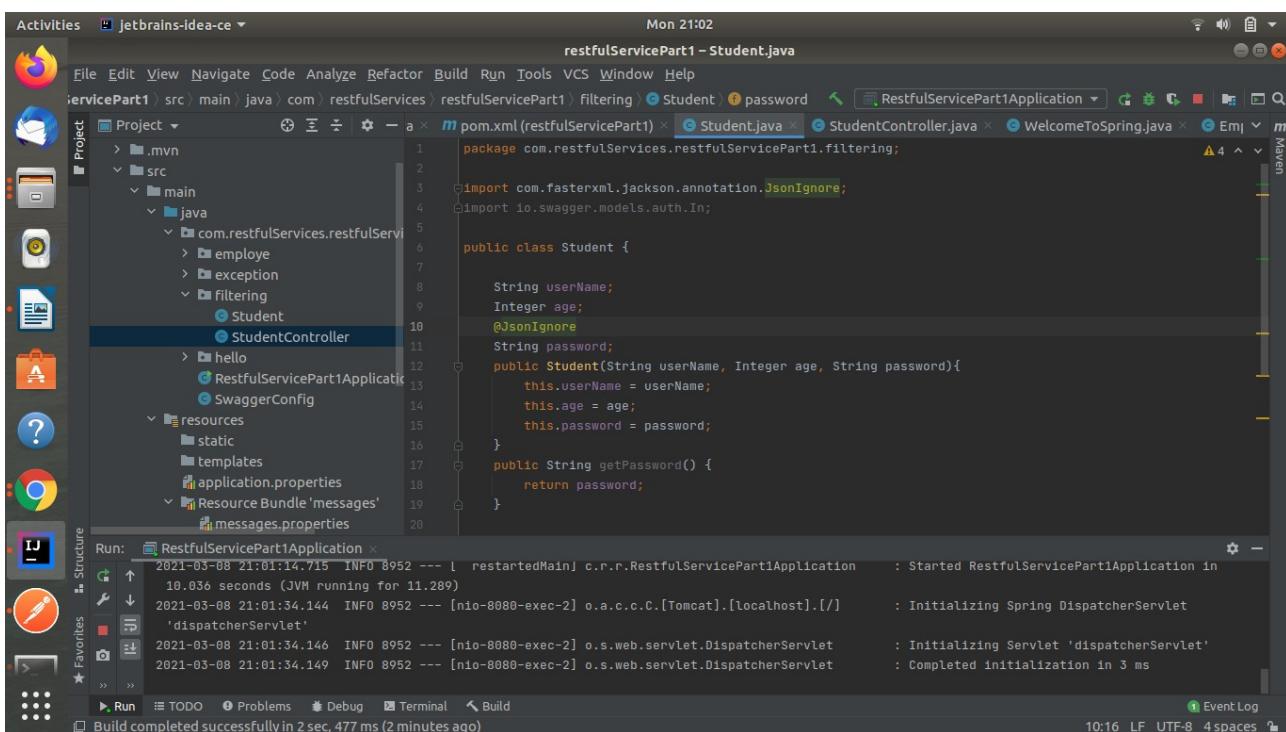
```
import org.springframework.web.bind.annotation.RestController;
import java.util.Arrays;
import java.util.List;

@RestController
public class StudentController {

    @GetMapping("/students")
    public Student getAllStudents(){
        return new Student( "NK563", 23, "nk789456");
    }

    @GetMapping("/student-list")
    public List<Student> getAllStudentList(){
        return Arrays.asList(new Student("mk56", 26, "hjvhvijv"),
                new Student("yk456", 18, "bshbsijv"));
    }
}
```

The run tab shows the application has started successfully.



The screenshot shows the IntelliJ IDEA interface with the code editor open to `Student.java`. The code defines a `Student` class with a `@JsonIgnore` annotation on the `password` field:

```
package com.restfulServices.restfulServicePart1.filtering;

import com.fasterxml.jackson.annotation.JsonIgnore;
import io.swagger.models.auth.In;

public class Student {

    String userName;
    Integer age;
    @JsonIgnore
    String password;

    public Student(String userName, Integer age, String password){
        this.userName = userName;
        this.age = age;
        this.password = password;
    }

    public String getPassword() {
        return password;
    }
}
```

The run tab shows the application has started successfully.

The screenshot shows the Postman application interface. In the top navigation bar, it says "Mon 21:02" and "Postman". The left sidebar has sections for "Collections", "APIs", "Environments", "Mock Servers", "Monitors", and "History". The main workspace is titled "My Workspace" and shows a message "No APIs yet". It includes a link to "Create an API". A search bar at the top right says "Search Postman". Below the search bar, there are four recent requests: "GET http://localhost:8080/students" (status 200 OK), "GET http://localhost:8080/students" (status 200 OK), "GET http://localhost:8080/students" (status 200 OK), and "GET http://localhost:8080/students" (status 200 OK). The current request is "GET http://localhost:8080/students". The "Params" tab is selected. The "Body" tab shows a JSON response:

```
1 {  
2   ... "userName": "NK563",  
3   ... "age": 23  
4 }
```

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

The screenshot shows the IntelliJ IDEA interface. The title bar says "Activities JetBrains-idea-ce" and "Mon 21:26". The central area displays the code for "StudentController.java" under the package "restfulServicePart1 - StudentController.java". The code contains two methods: "getAllStudents()" and "getStudentList()". The "getStudentList()" method uses a "SimpleBeanPropertyFilter" to filter out all fields except "userName". The "Run" tab at the bottom shows the application's startup logs:

```
2021-03-08 21:24:37.319 INFO 9950 --- [ restartedMain] c.r.r.RestfulServicePart1Application : Started RestfulServicePart1Application in 9.177 seconds (JVM running for 10.333)  
2021-03-08 21:24:39.571 INFO 9950 --- [nio-8080-exec-1] o.a.c.c.c.Tomcat@localhost:[/] : Initializing Spring DispatcherServlet  
2021-03-08 21:24:39.572 INFO 9950 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'  
2021-03-08 21:24:39.585 INFO 9950 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 13 ms
```

The screenshot shows the IntelliJ IDEA interface. The code editor displays the `Student.java` file, which contains Java code for a `Student` class. The `pom.xml` file is also visible in the project tree. The `Run` tab shows the application logs for `RestfulServicePart1Application`, indicating successful startup and initialization of the Spring DispatcherServlet.

```

import com.fasterxml.jackson.annotation.JsonIgnore;
import io.swagger.models.auth.In;
@JsonFilter("StudentFilter")
public class Student {
    String userName;
    Integer age;
    //JsonIgnore //un-comment it for static filtering
    String password;
    public Student(String userName, Integer age, String password){
        this.userName = userName;
        this.age = age;
        this.password = password;
    }
    public String getPassword() {
        return password;
    }
    public Integer getAge() {
        return age;
    }
}

```

The screenshot shows the Postman interface. A GET request is made to `http://localhost:8080/student-list`. The response body is displayed in Pretty format, showing a JSON array of two objects representing student data.

```

[{"id": 1, "name": "mk56", "age": 20}, {"id": 2, "name": "yk456", "age": 22}

```

## \*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

Mon 23:47

Postman

File Edit View Help

Home Workspaces Reports Explore

My Workspace

New Import

GET http://localhost... ● GET http://localhost... ● GET http://localhost... ● GET http://localhost... ● + ... No Environment

<http://localhost:8080//person/param?version=1>

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> version	1			
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "name": "Nitin"
3

```

200 OK 294 ms 180 B Save Response

Find and Replace Console Bootcamp Runner Trash

Mon 23:47

Postman

File Edit View Help

Home Workspaces Reports Explore

My Workspace

New Import

GET http://localhost... ● GET http://localhost... ● GET http://localhost... ● GET http://localhost... ● + ... No Environment

<http://localhost:8080//person/param?version=2>

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> version	2			
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "name": {
3     "firstName": "Nitin",
4     "lastName": "khandelwal"
5   }
6

```

200 OK 19 ms 218 B Save Response

Find and Replace Console Bootcamp Runner Trash

Activities Postman

Tue 00:01 Postman

File Edit View Help

Home Workspaces Reports Explore

New Import

My Workspace

No APIs yet

APIs define related collections and environments under a consistent schema.

Create an API

GET http://localhost:8080/person/header

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers (9)

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> X-API-VERSION	1			
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
```

200 OK 401 ms 180 B Save Response

Find and Replace Console Bootcamp Runner Trash

The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace is titled 'My Workspace' and displays a message 'No APIs yet'. In the center, a request card is open for 'http://localhost:8080/person/header'. The method is 'GET', and the URL is 'http://localhost:8080/person/header'. The 'Headers' tab is selected, showing a single header 'X-API-VERSION' with the value '1'. Below the headers, the 'Body' tab is selected, showing a JSON response with a single key 'name' and the value 'Nitin'. The response status is '200 OK' with a duration of '401 ms' and a size of '180 B'. At the bottom, there are buttons for 'Find and Replace' and 'Console', along with links for 'Bootcamp', 'Runner', 'Trash', and help.

Activities Postman

Tue 00:01 Postman

File Edit View Help

Home Workspaces Reports Explore

New Import

My Workspace

No APIs yet

APIs define related collections and environments under a consistent schema.

Create an API

GET http://localhost:8080/person/header

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers (9)

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> X-API-VERSION	2			
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
```

200 OK 16 ms 218 B Save Response

Find and Replace Console Bootcamp Runner Trash

The screenshot shows the Postman application interface, identical to the one above but with a different header value. The 'X-API-VERSION' header is now set to '2'. The response body is a more complex JSON object with three nested levels: 'name' (which contains 'firstName' and 'lastName'). The rest of the interface, including the request URL, tabs, and status details, remains the same.

Tue 00:37

Postman

File Edit View Help

Home Workspaces Reports Explore

New Import

GET http://l... ● GET http://l... ● GET http://l... ● GET http://l... ● GET http://l... ●

No Environment

http://localhost:8080/person/produces?Accept=application/vnd.company.app-v1+json

GET http://localhost:8080/person/produces?Accept=application/vnd.company.app-v1+json

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Accept	application/vnd.company.app-v1+json			
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 ... "name": "Nitin"
2 ...
3 ...
```

200 OK 11 ms 199 B Save Response

Find and Replace Console Bootcamp Runner Trash

Activities Postman

My Workspace Collections APIs Environments Mock Servers Monitors History

No APIs yet APIs define related collections and environments under a consistent schema. Create an API

This screenshot shows the Postman application interface. The left sidebar displays various workspace options like Collections, APIs, and Environments. The main workspace is titled 'My Workspace' and shows a message 'No APIs yet'. A central panel displays a POST request to 'http://localhost:8080/person/produces?Accept=application/vnd.company.app-v1+json'. The 'Headers' tab is selected, containing a single entry 'Accept: application/vnd.company.app-v1+json'. The 'Body' tab shows a JSON response with one key, 'name', with the value 'Nitin'. The status bar at the bottom indicates a 200 OK response with 11 ms latency and 199 B size.

Tue 00:44

Postman

File Edit View Help

Home Workspaces Reports Explore

New Import

GET http://l... ● GET http://l... ● GET http://l... ● GET http://l... ● GET http://l... ●

No Environment

http://localhost:8080/person/produces

GET http://localhost:8080/person/produces

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Headers < 6 hidden

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Accept	application/vnd.company.app-v2+json				
Key	Value	Description			

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
```

200 OK 267 ms 237 B Save Response

Find and Replace Console Bootcamp Runner Trash

Activities Postman

My Workspace Collections APIs Environments Mock Servers Monitors History

No APIs yet APIs define related collections and environments under a consistent schema. Create an API

This screenshot shows the Postman application interface. The left sidebar displays various workspace options like Collections, APIs, and Environments. The main workspace is titled 'My Workspace' and shows a message 'No APIs yet'. A central panel displays a POST request to 'http://localhost:8080/person/produces'. The 'Headers' tab is selected, containing a single entry 'Accept: application/vnd.company.app-v2+json'. The 'Body' tab shows a JSON response with a nested object 'name' containing 'firstName' and 'lastName'. The status bar at the bottom indicates a 200 OK response with 267 ms latency and 237 B size.

Activities Postman

Tue 00:45

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

GET http://l... ● No Environment

Save Invitations Notifications

Upgrade

My Workspace

New Import

Collections +

No APIs yet

APIs

Environments

Mock Servers

Monitors

History

Create an API

http://localhost:8080/person/produces

GET http://localhost:8080/person/produces

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Headers 6 hidden

KEY VALUE DESCRIPTION Bulk Edit Presets

Accept application/vnd.company.app-v1+json

Key Value Description

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

200 OK 318 ms 199 B Save Response

Find and Replace Console

Bootcamp Runner Trash

Activities JetBrains-Idea-ce

Tue 11:56

restfulServicePart1 - VersionController.java

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

src main java com restfulServices restfulServicePart1 versioning VersionController produces1 RestfulServicePart1Application

Project

exception CustomizedResponseEntity ExceptionResponse

filtering Student StudentController

hello

versioning Name Person1 Person2

VersionController RestfulServicePart1Application SwaggerConfig

resources static templates application.properties

Resource Bundle 'messages'

messages.properties messages\_de.properties messages\_sv.properties

test target .gitignore

VersionController.java EmployeeDaoService.java Employee.java StudentController.java WelcomeTo.java

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class VersionController {
    //URI versioning
    @GetMapping("v1/person")
    public Person1 person1(){
        return new Person1(name: "Nitin");
    }
    @GetMapping("v2/person")
    public Person2 person2(){
        return new Person2(new Name("Nitin", "khandelwal"));
    }

    //Request Parameter versioning
    @GetMapping(value = "/person/param", params = "version=1")
    public Person1 param1(){
        return new Person1(name: "Nitin");
    }
    @GetMapping(value = "/person/param", params = "version=2")
    public Person2 param2(){
        return new Person2(new Name("Nitin", "khandelwal"));
    }

    //Custom Header Versioning
    @GetMapping(value = "/person/header", headers = "X-API-VERSION=1")
}
```

Run: RestfulServicePart1Application

Event Log

Build completed successfully in 2 sec, 453 ms (today 10:39 AM)

39:94

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities, Jetbrains-idea-ce, Tue 11:56, restfulServicePart1 – VersionController.java
- Project Structure:** Shows a hierarchical view of the project structure, including packages like exception, filtering, hello, and versioning, and files like EmployeeDaoService.java, Employee.java, StudentController.java, and WelcomeTo.java.
- Code Editor:** Displays the `VersionController.java` file with annotations for different API versions (Customized ResponseEntity, @GetMapping, @GetMapping, @Produces). The code defines methods for returning Person objects based on the requested headers or version.
- Bottom Bar:** Run, TODO, Problems, Debug, Terminal, Build, Event Log, and a build status message: "Build completed successfully in 2 sec, 453 ms (today 10:39 AM)"

## \*HATEOAS

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

The screenshot shows the Postman interface with the following details:

- Title Bar:** Activities, Postman, Tue 00:47
- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Request Panel:** A GET request to `http://localhost:8080/employees/1`. The Headers tab is selected, showing a header `X-API-VERSION` with value `2`.
- Body Panel:** Shows the response body in Pretty format, which includes an ID, name, age, and a link to the collection resource (`http://localhost:8080/employees`).
- Status Bar:** 200 OK, 169 ms, 269 B, Save Response.

Activities Jetbrains-idea-ce Tue 00:48

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

restfulServicePart1 – EmployeeResource.java

Project src main java com restfulServices restfulServicePart1 employe EmployeeResource getEmp RestfulServicePart1Application EmployeeDaoService StudentController.java WelcomeToSpring.java EmployeeResource.java

```
EmployeeResource.java
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

EmployeeResource.java

```
@GetMapping(path = "/employees")
public List<Employee> getAllEmp() { return service.findAll(); }

@GetMapping(path = "/employees/{id}")
public EntityModel<Employee> getEmp(@PathVariable int id){
    Employee emp = service.findOne(id);
    if(emp == null) throw new EmployeeNotFoundException("id- " + id);

    EntityModel<Employee> resource = EntityModel.of(emp);
    WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).getAllEmp());
    resource.add(linkTo.withRel("all-employees"));
    return resource;
}

@DeleteMapping(path = "/employees/{id}")
public void deleteEmp(@PathVariable int id){
    Employee emp = service.delete(id);
    if(emp.getId() == null) throw new EmployeeNotFoundException("id- " + id);
}

@PutMapping("/employees")
```

Run: RestfulServicePart1Application

```
2021-03-09 00:45:09.214 INFO 14454 --- [ restartedMain c.r.R.RestfulServicePart1Application : Started RestfulServicePart1Application in 7.836 seconds (JVM running for 8.847)
2021-03-09 00:45:14.282 INFO 14454 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet
'dispatcherServlet'
2021-03-09 00:45:14.282 INFO 14454 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-03-09 00:45:14.287 INFO 14454 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
```

Build completed successfully in 1 sec, 960 ms (3 minutes ago)

Event Log