MUSKAN LAMA
6648
GIT HUB LINK

Internationalization

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

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

Employee .java BEAN class

```
package com.TTN.restfulday2.internationalizationHateos;
import org.springframework.hateoas.EntityModel;
//my bean class
public class Employee extends EntityModel<Employee> {
   private Integer id;
   private String userName;
   private Integer age;
   //constructor
   public Employee(Integer id, String name, Integer age) {
       this.id = id;
       this.userName = name;
       this.age = age;
   }
   public Integer getId() {
       return id;
   }

   public void setId(Integer id) {
       this.id = id;
   }
   public String getUserName() {
       return userName;
   }
   public void setName(String name) {
       this.userName = name;
   }
   public Integer getAge() {
       return age;
   }
 public void setAge(Integer age) {
```

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

Controller class

```java
package com.TTN.restfulday2.internationalizationHateos;

import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.*;

import org.springframework.hateoas.EntityModel;
import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;
import com.TTN.restfulday2.exception.IdNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.net.URI;
import java.util.List;
import java.util.Locale;



@RestController
public class Controller {


   private MessageSource messageSource;


   @Autowired
   private Service service;


   public Controller(MessageSource messageSource)
   {
       this.messageSource=messageSource;
   }

   //using get mapping getting all list of employees
```

```
    @GetMapping("/users")
    public List<Employee> getAll()
    {

        return service.findAll();
    }

    //getting Pathvariable
    //Implement GET
    // http request using path variable top get one employee
    //HATEOS
    @GetMapping("/users/{id}")
    public EntityModel<Employee> retrieveUser(@PathVariable int id) throws
IdNotFoundException {
        Employee employee = service.findOne(id);

        if(employee==null)
            throw new IdNotFoundException("id:"+id);

        EntityModel<Employee> entityModel = EntityModel.of(employee);

        WebMvcLinkBuilder link =  linkTo(methodOn(this.getClass()).getAll());
        entityModel.add(link.withRel("all-users"));

        return entityModel;
    }

     //apply Internationalization allowing message to display in english
    @GetMapping(path="/hello")
   public String helloNameInterNationalized()
   {
       //requestheader
       //using locale to fetch from messages.properties

        Locale locale= LocaleContextHolder.getLocale();
      return messageSource.getMessage("hello.name.message",null,"Default
message",locale);
   }
```

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

```
   @GetMapping(path="/hello/{username}")
    public String localized(@PathVariable String username)
   {
      //using locale to fetch from messages.properties
      Locale locale2=LocaleContextHolder.getLocale();
      //creating array of strings to pass as a parameter
       String name[] = new String[]{username};
//passing name as argument
      return  messageSource.getMessage("userName",name,"Default
message",locale2);


   }
```

```
}
```

Service.java

```
package com.TTN.restfulday2.internationalizationHateos;

import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;


//methods

//managed by spring and it is autowired in Employee resource
@Component
public class Service {


   //list
   private static List<Employee> employees = new ArrayList<>();


   //setting up values of Employee bean
   static
   {
       employees.add(new Employee(12,"muskan",23));
       employees.add(new Employee(13,"sofi",22));


   }

   //returns all Employee data using List in json format
   public List<Employee> findAll()
   {
       return employees;
   }

   //method which will match and return the employee details according to id
   public Employee findOne(int id)
   {

       //functional programming
       Predicate<? super Employee> predicate =
               employee -> employee.getId()==(id);


       //concerting List to stream
       //orElse if id is not found to handle exception
       //but no response will come
       return employees.stream().filter(predicate).findFirst().orElse(null);
   }

   public Employee findUser(String userName)
   {

       //functional programming
       Predicate<? super Employee> predicate =
               employee -> employee.getUserName().equals(userName);
```

```
        //concerting List to stream
        //orElse if id is not found to handle exception
        //but no response will come
        return employees.stream().filter(predicate).findFirst().orElse(null);
    }


}
```

messages.properties

```
#default english message
hello.name.message=hello

#passing parameter in message.properties
userName=Hello {0}
```

messages-de.properties

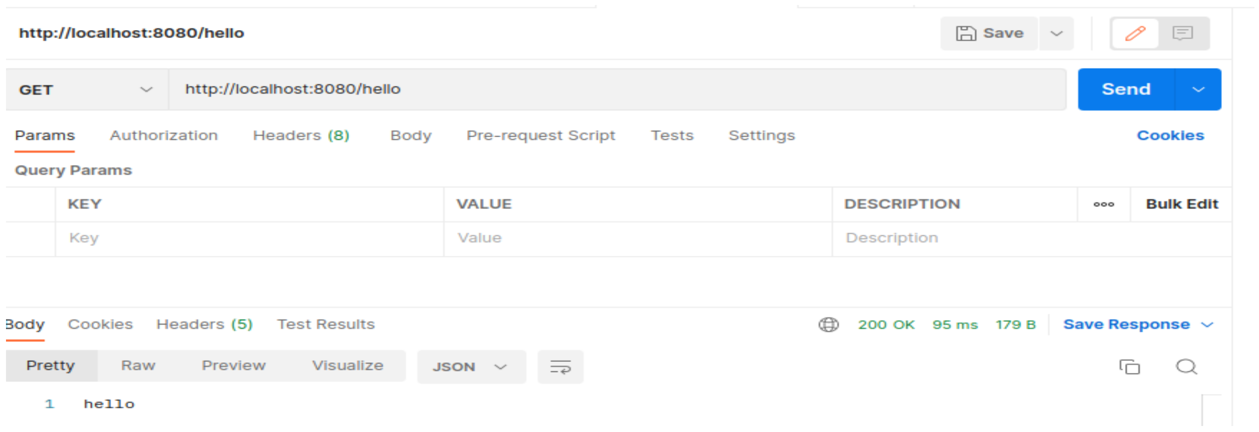
```
hello.name.message=hallo
```

messages-sv.properties

```
hello.name.message=Hallå
```

OUTPUT

English



http://localhost:8080/hello

GET http://localhost:8080/hello

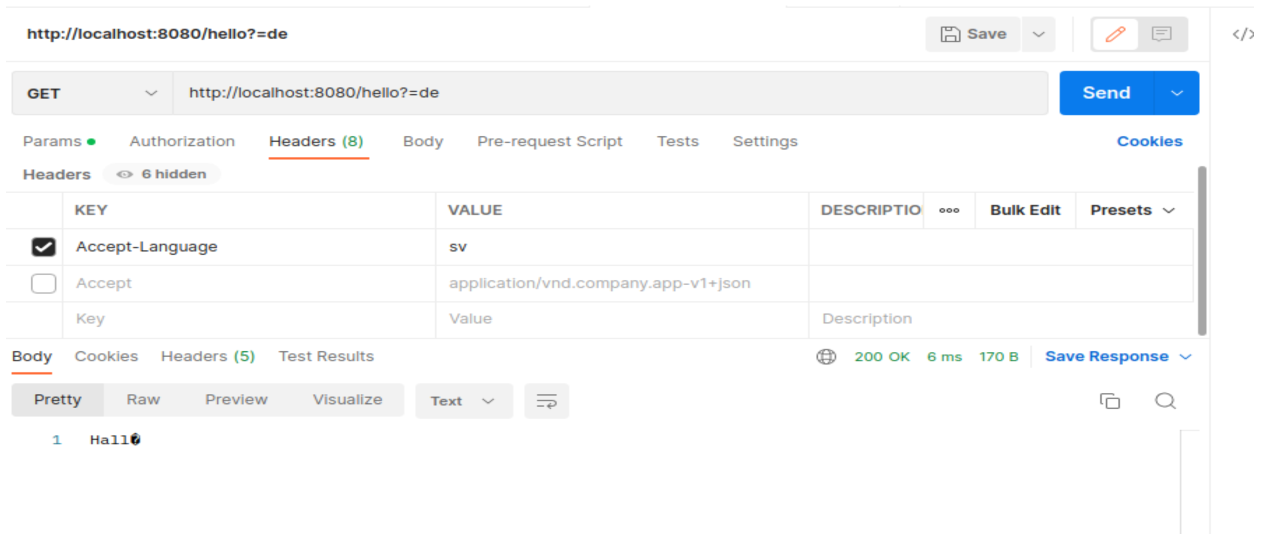
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

200 OK 95 ms 179 B

```
hello
```

SWEDEN



http://localhost:8080/hello?=de

GET http://localhost:8080/hello?=de

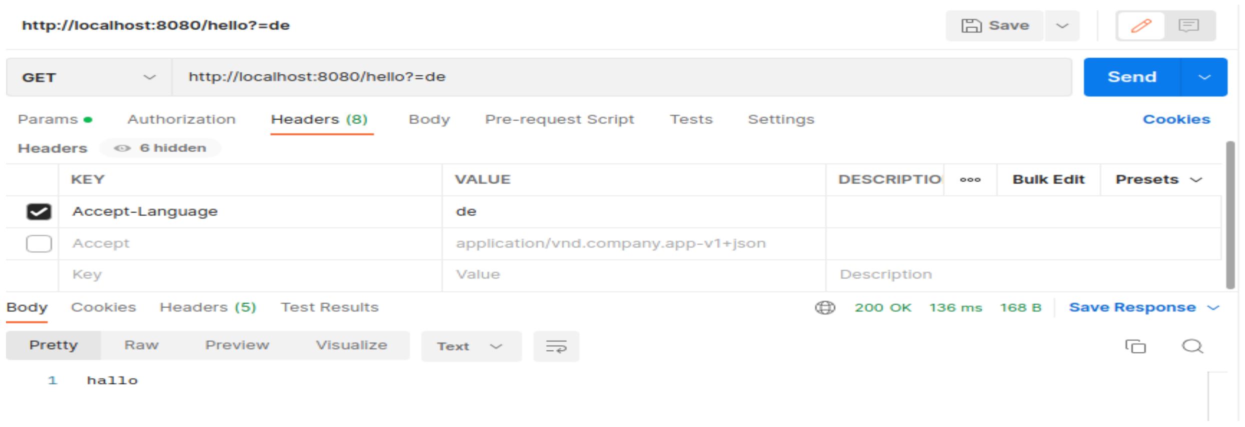
Headers 6 hidden

	KEY	VALUE	DESCRIPTIO
✓	Accept-Language	sv	
	Accept	application/vnd.company.app-v1+json	
	Key	Value	Description

200 OK 6 ms 170 B

```
Hall�
```

GERMAN



http://localhost:8080/hello?=de

GET http://localhost:8080/hello?=de

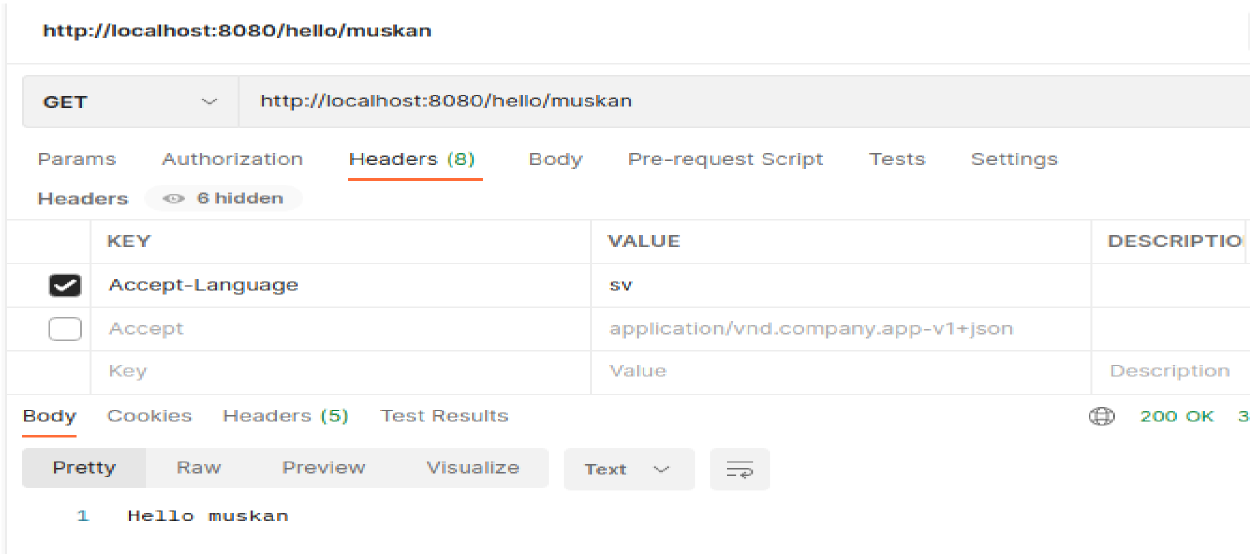
Headers 6 hidden

	KEY	VALUE	DESCRIPTIO
✓	Accept-Language	de	
	Accept	application/vnd.company.app-v1+json	
	Key	Value	Description

200 OK 136 ms 168 B

```
hallo
```

Passing parameters in messages.properties


http://localhost:8080/hello/muskan

GET http://localhost:8080/hello/muskan

	KEY	VALUE	DESCRIPTIO
✓	Accept-Language	sv	
	Accept	application/vnd.company.app-v1+json	
	Key	Value	Description

```
Hello muskan
```

*Content Negotiation

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

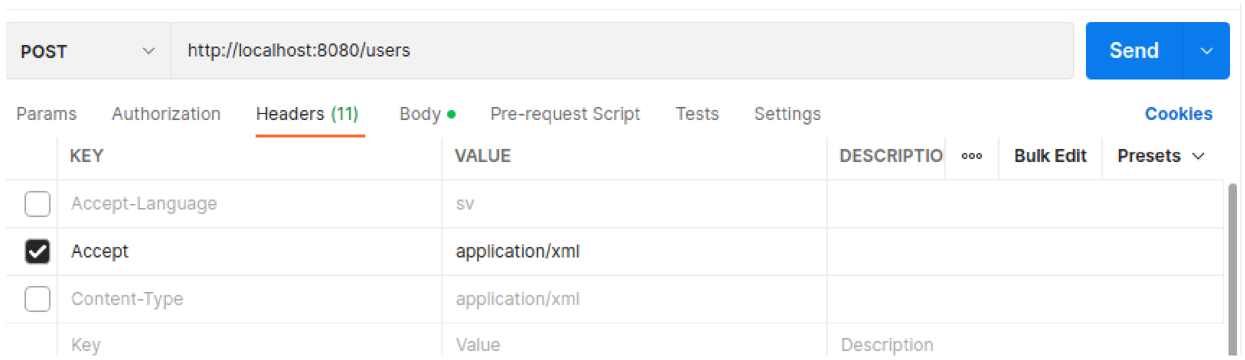
4. Create GET Method to fetch the list of users in XML format

POST METHOD

```
//Apply validation while

// create a new employee using POST http Request.

// create user

@PostMapping(path="/users"  ,produces = "application/xml ",consumes =
"application/xml")
public ResponseEntity<Employee> createUser(@RequestBody Employee employee)
//requestBody for creating objects of json data
{
  Employee save= service.save(employee);

   URI location= ServletUriComponentsBuilder.fromCurrentRequest()
           .path("/{age}")
           .buildAndExpand(save.getAge())
           .toUri();
   // to return the uri location  of the created object
```

```
    return ResponseEntity.created(location).build(); // created and return the
uri
}
```

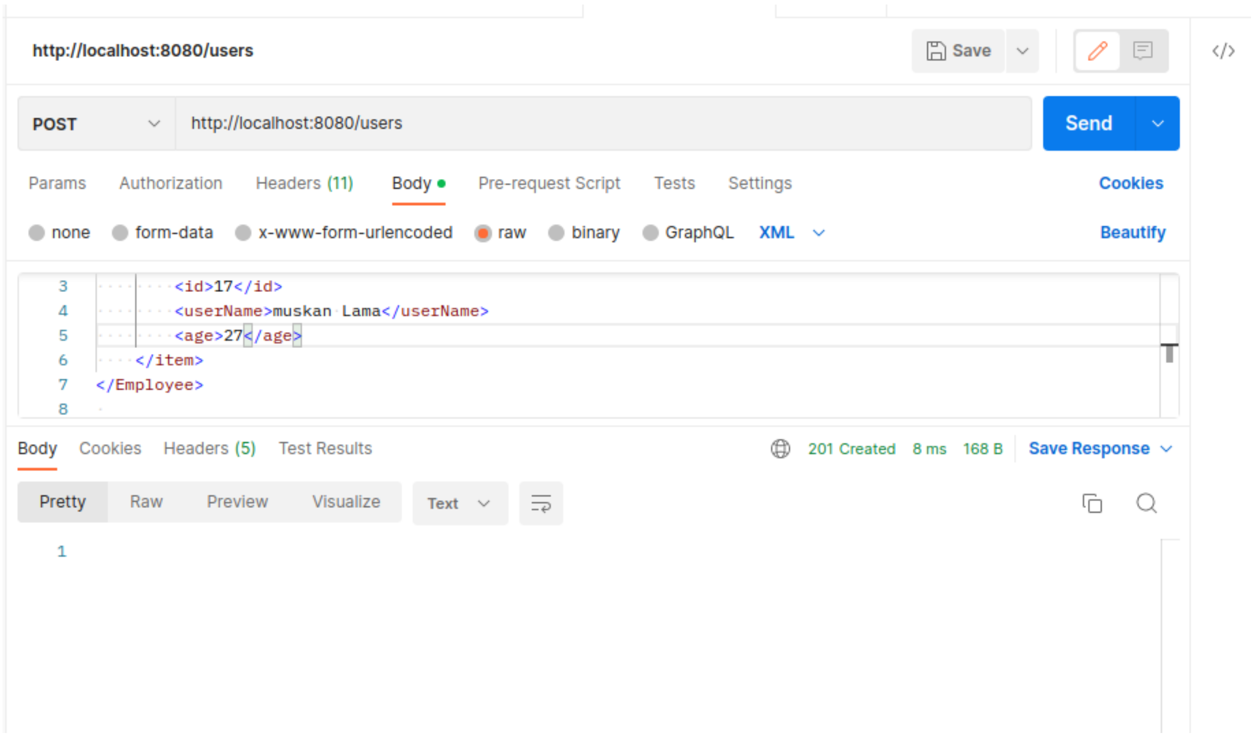
Output



POST http://localhost:8080/users

	KEY	VALUE	DESCRIPTION
☐	Accept-Language	sv	
☑	Accept	application/xml	
☐	Content-Type	application/xml	
	Key	Value	Description

Using accept



POST http://localhost:8080/users

Body: raw, XML

```
        <id>17</id>
        <userName>muskan Lama</userName>
        <age>27</age>
    </item>
</Employee>
```

201 Created 8 ms 168 B

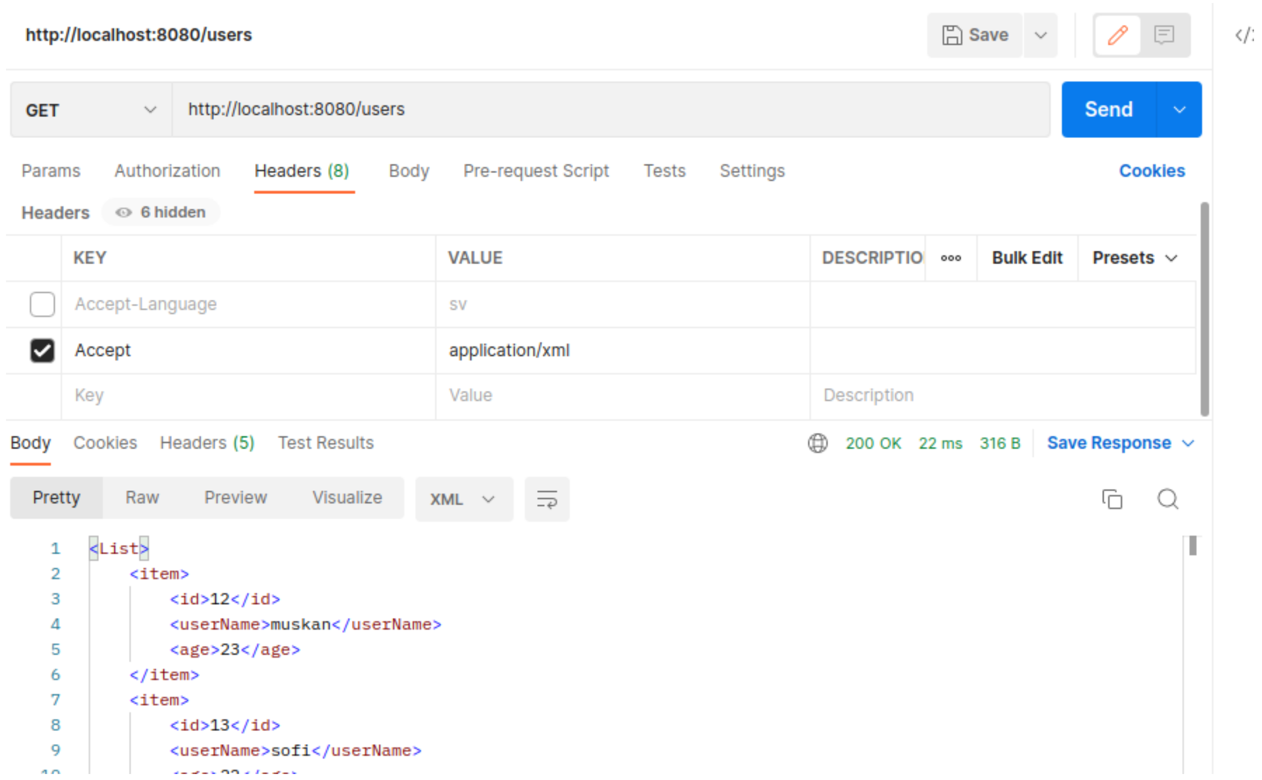
Get

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

```
//getting Pathvariable
//Implement GET
```

```java
// http request using path variable top get one employee
//HATEOS
//content negotiation
@GetMapping("/users/{id}")
public EntityModel<Employee> retrieveUser(@PathVariable int id) throws
IdNotFoundException {
   Employee employee = service.findOne(id);

   if(employee==null)
       throw new IdNotFoundException("id:"+id);

   EntityModel<Employee> entityModel = EntityModel.of(employee);


   WebMvcLinkBuilder link =  linkTo(methodOn(this.getClass()).getAll());

   entityModel.add(link.withRel("all-users"));

   return entityModel;
}
```

Output



	KEY	VALUE	DESCRIPTIO
☐	Accept-Language	sv	
☑	Accept	application/xml	
	Key	Value	Description

```xml
<List>
    <item>
        <id>12</id>
        <userName>muskan</userName>
        <age>23</age>
    </item>
    <item>
        <id>13</id>
        <userName>sofi</userName>
```

***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.

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.

SpringConfig.java (to configure which path to document and all)

```
package com.TTN.restfulday2.swagger;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import static springfox.documentation.builders.PathSelectors.regex;

@Configuration
@EnableSwagger2
public class SpringConfig {
   @Bean
   public Docket api() {

           return new Docket(DocumentationType.SWAGGER_2)
                   .select()
//selecting the base package
.apis(RequestHandlerSelectors.basePackage("com.TTN.restfulday2.swagger"))
                   .paths(PathSelectors.any())
//select of the path
                   .build();
       }
   }
```

Swagger.java bean

```
{

   private String name;
   private Integer age;


   public Swagger(String name, Integer age) {
       this.name = name;
       this.age = age;
   }

   public String getName() {
       return name;
   }

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

   public Integer getAge() {
       return age;
   }

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

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

Controller.java

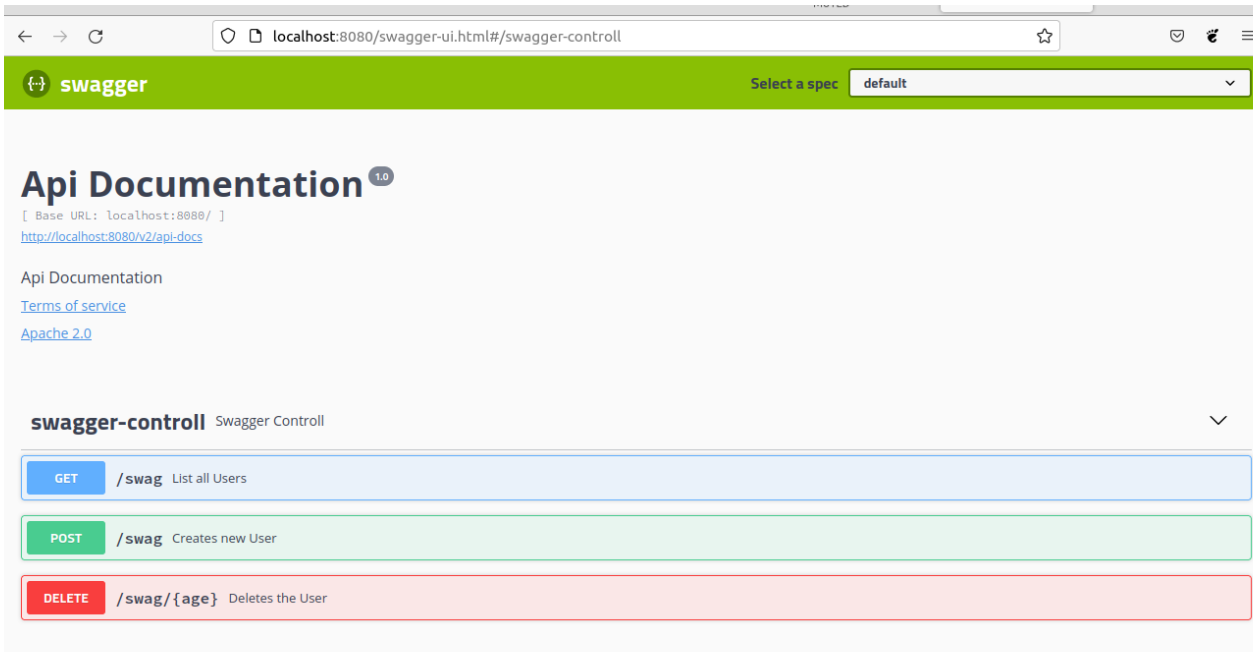
```
package com.TTN.restfulday2.swagger;

import java.net.URI;

import io.swagger.annotations.ApiOperation;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.net.URI;
import java.util.List;

@RestController
public class SwaggerControll {


   @Autowired
   private SwaggerResource swaggerResource;
   public SwaggerControll(SwaggerResource swaggerResource) {

       this.swaggerResource = swaggerResource;
   }


   //getting all the Employee
   //mapping to specific url using GET
   @GetMapping( path="/swag")
   @ApiOperation(value = "List all Users",
           notes = "List all Users using it's service method", response =
Swagger.class)
   public List<Swagger> getAll()
   {

       return swaggerResource.findAll();
   }




   //DELETE
   @DeleteMapping(path="/swag/{age}")
   @ApiOperation(value = "Deletes the User",
           notes = "Deletes the user associated with the passed AGE", response
= Swagger.class)
   public void delete(@PathVariable int age)
   {


       //deleting
      swaggerResource.delete(age);

   }


   //Apply validation while
   // create a new employee using POST http Request.
   // create user
   @PostMapping(path="/swag")
   @ApiOperation(value = "Creates new User",
           notes = "Creates new user using it's service method", response =
Swagger.class)
   public ResponseEntity<Swagger> createUser(@RequestBody Swagger swagger)
   //requestBody for creating objects of json data
   {
       Swagger save= swaggerResource.save(swagger);
```

```
        URI location= ServletUriComponentsBuilder.fromCurrentRequest()
                .path("/{age}")
                .buildAndExpand(save.getAge())
                .toUri();
        // to return the uri location  of the created object

        return ResponseEntity.created(location).build(); // created and return
the uri
   }

}
```

SwaggerResource.java

```
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;

@Component
public class SwaggerResource
   {


        //list
        private static List<Swagger> swaggers = new ArrayList<>();


        //setting up values
        static
        {
            swaggers.add(new Swagger("muskan", 23));
            swaggers.add(new Swagger("sofi", 22));


        }


        //returns all Employee data using List in json format
        public List<Swagger> findAll()
        {
            return swaggers;
        }


        // and return it and display in json
        public Swagger save (Swagger swagger)
        {
            swaggers.add(swagger);
            return swagger;
        }

        //method to delete
        public void delete ( int age)
        {
            //deleting by id
            swaggers.removeIf(e -> e.getAge().equals(age));

        }


}
```

Output



localhost:8080/swagger-ui.html#/swagger-controll

Api Documentation 1.0

[Base URL: localhost:8080/]

http://localhost:8080/v2/api-docs

Api Documentation

Terms of service

Apache 2.0

swagger-controll Swagger Controll

GET /swag List all Users

POST /swag Creates new User

DELETE /swag/{age} Deletes the User

***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)

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

FilteringController.java

```
package com.TTN.restfulday2.staticAndDynamicFiltering;
import com.TTN.restfulday2.internationalizationHateosCN.Service;
import com.fasterxml.jackson.databind.ser.FilterProvider;
import com.fasterxml.jackson.databind.ser.impl.SimpleBeanPropertyFilter;
import com.fasterxml.jackson.databind.ser.impl.SimpleFilterProvider;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.converter.json.MappingJacksonValue;
import org.springframework.stereotype.Component;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@Component
public class FilteringController {


   @Autowired
  private Service service;

   //static filtering
   @GetMapping("/filtering")
   public UserDetails filtering(UserDetails userDetails)
   {

       return  new UserDetails("value1",12,"value3");

   }
   //dynamic filtering
   @GetMapping("/filtering-dynamic")
   public MappingJacksonValue filteringDynamic(UserDetails userDetails)

   {
       UserDetails userDetails3= new UserDetails("value2",34,"value4");
       MappingJacksonValue mappingJacksonValue=new
MappingJacksonValue(userDetails3);
       SimpleBeanPropertyFilter filter =
               SimpleBeanPropertyFilter.filterOutAllExcept("name","age");
       FilterProvider filters =
               new SimpleFilterProvider().addFilter("SomeBeanFilter", filter
);

       mappingJacksonValue.setFilters(filters );

       return mappingJacksonValue;

   }
}
```

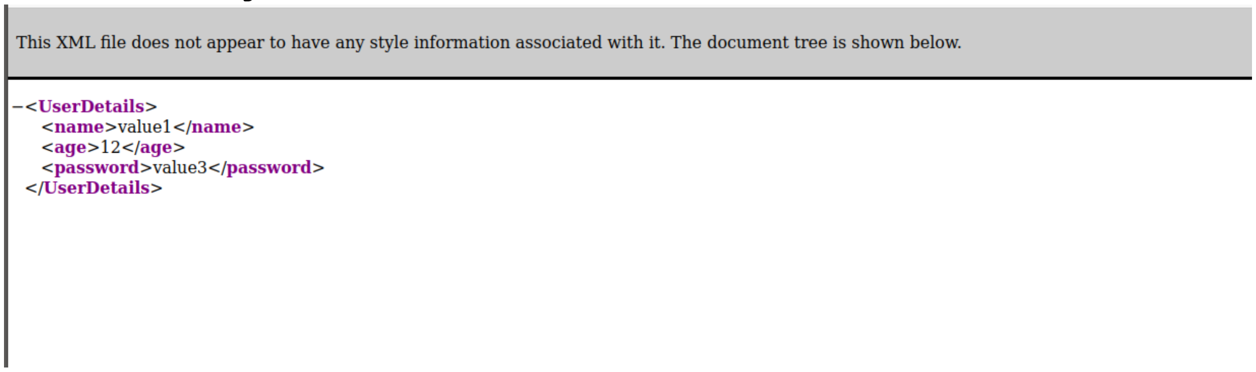
ServiceSD.class
```java
package com.TTN.restfulday2.staticAndDynamicFiltering;

import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;


//methods

//managed by spring and it is autowired in Employee resource
@Component
public class ServiceSD {


   //list
   private static List<UserDetails> userDetailsList = new ArrayList<>();


   //setting up values of Employee bean
   static {
       userDetailsList.add(new UserDetails("Muskan", 22, "3rewr"));
       userDetailsList.add(new UserDetails("Geetanjali", 32, "3sjd"));


   }


}
```

UserDetails.java
```java
import net.minidev.json.annotate.JsonIgnore;
import org.springframework.stereotype.Component;

@JsonFilter("SomeBeanFilter")
public class UserDetails {

   @JsonIgnore
   private String name;
   private Integer age;
   private String password;

   public UserDetails(String name, Integer age, String password) {
       this.name = name;
       this.age = age;
       this.password = password;
   }

   public String getName() {
       return name;
   }

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

   public Integer getAge() {
       return age;
   }

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

```
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

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

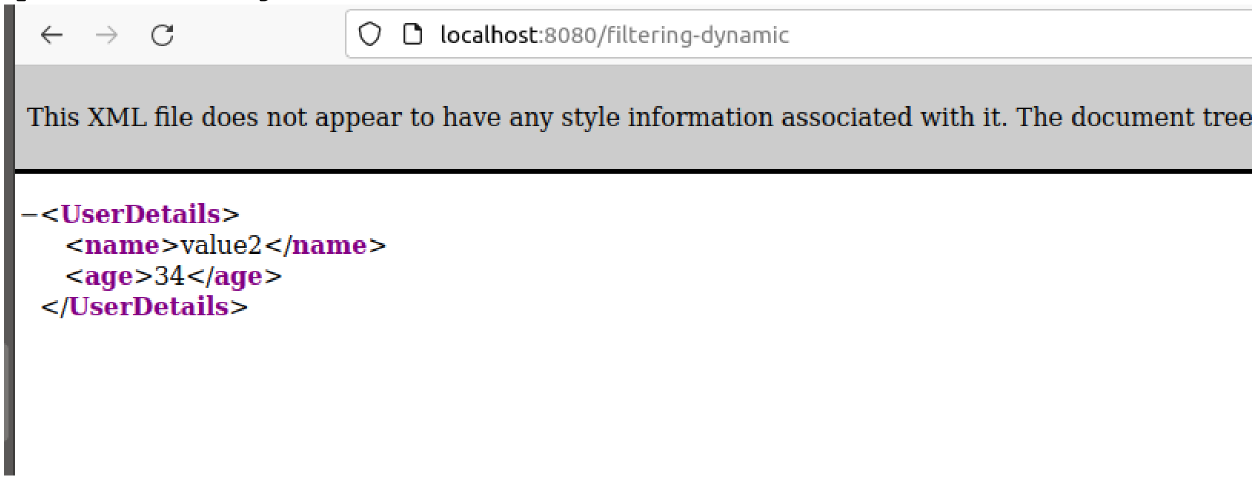
Output

Static filtering



```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

-<UserDetails>
   <name>value1</name>
   <age>12</age>
   <password>value3</password>
 </UserDetails>
```

Dynamic filtering



localhost:8080/filtering-dynamic

```
This XML file does not appear to have any style information associated with it. The document tree

-<UserDetails>
   <name>value2</name>
   <age>34</age>
 </UserDetails>
```

*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
-

UserV1.java

```java
package com.TTN.restfulday2.versioning;

public class UserV1 {
   private String name;

   @Override
   public String toString() {
       return "UserV1{" +
               "name='" + name + '\'' +
               '}';
   }

   public String getName() {
       return name;
   }

   public UserV1(String name) {
       this.name = name;
   }
}
```

UserV2.java

```java
package com.TTN.restfulday2.versioning;

public class UserV2 {


   private String firstName;
   private String lastName;

   public UserV2(String firstName, String lastName) {
       this.firstName = firstName;
       this.lastName = lastName;
   }

   public String getFirstName() {
       return firstName;
   }

   public void setFirstName(String firstName) {
       this.firstName = firstName;
   }

   public String getLastName() {
       return lastName;
   }

   public void setLastName(String lastName) {
       this.lastName = lastName;
   }

   @Override
   public String toString() {
       return "UserV2{" +
               "firstName='" + firstName + '\'' +
               ", lastName='" + lastName + '\'' +
               '}';
   }
```

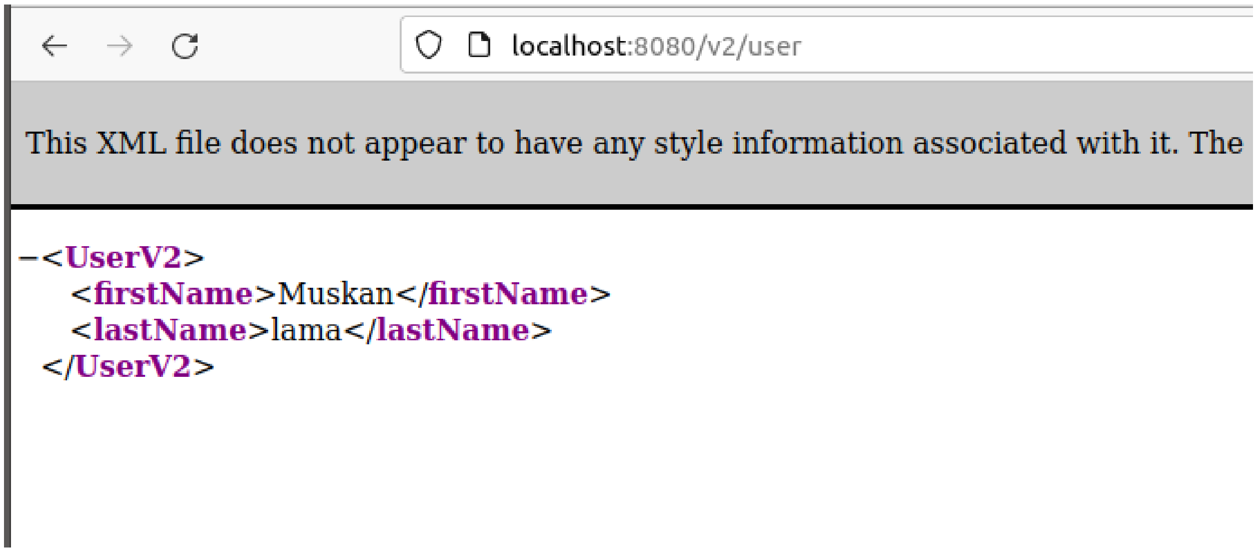
VersionController.java

```java
ackage com.TTN.restfulday2.versioning;

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

@RestController
public class VersionController {


   //uri
   @GetMapping("v1/user")
   public UserV1 userV1() {
       return new UserV1("muskan");
   }

   @GetMapping("v2/user")
   public UserV2 userV2() {
       return new UserV2("Muskan", "lama");

   }

   //params
```

```
    @GetMapping(path = "/user", params = "version=1")
    public UserV1 requestParameterVersioning1() {
        return new UserV1("Muskan");
    }


    @GetMapping(path = "/user", params = "version=2")
    public UserV2 requestParameterVersioning2() {
        return new UserV2("Muskan", "lama");
    }


   //headres cutom request headers
    @GetMapping(path = "/user/head",headers="X-API-VERSION=1")
    public UserV1 requestHeader1() {
        return new UserV1("Muskan");
    }

    @GetMapping(path = "/user/head", headers="X-API-VERSION=2")
    public UserV2 requestHeader2() {
        return new UserV2("Muskan", "lama");
    }

//media type nversioning
    @GetMapping(path = "/user/accept", produces =
"application/vnd.company.app-v1+json")
    public UserV2 requestMIME() {
        return new UserV2("Muskan", "lama");
    }

    @GetMapping(path = "/user/accept", produces =
"application/vnd.company.app-v2+json")
    public UserV1 requestMIME2() {
        return new UserV1("Muskan");
    }




}
```

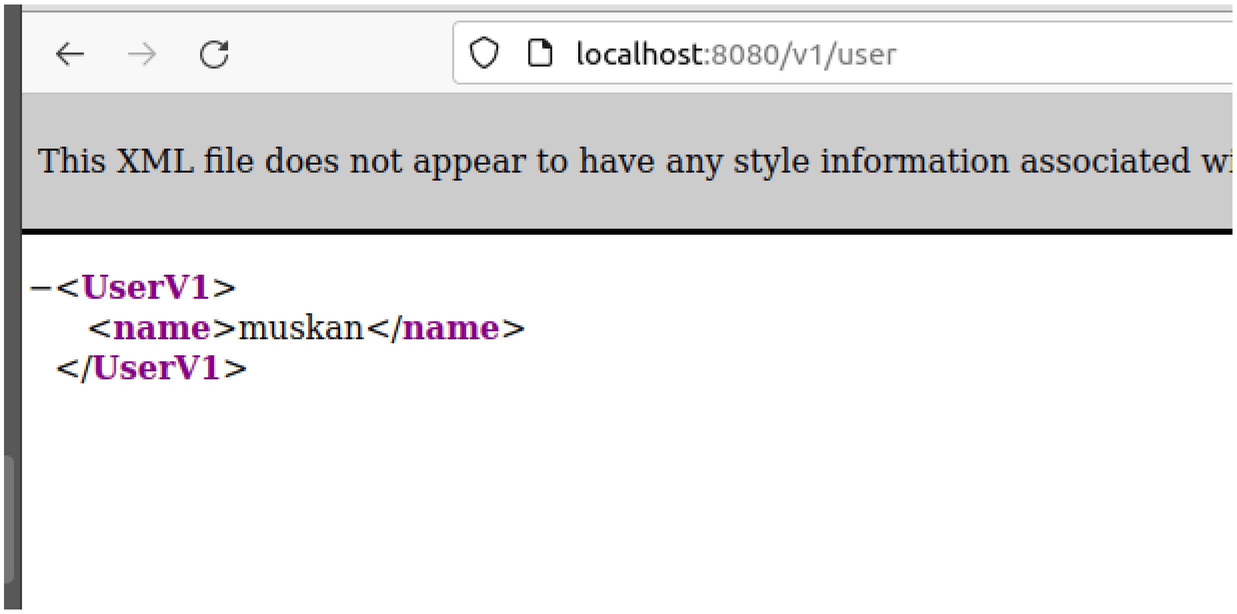
OUTPUT

Uri mapping


localhost:8080/v2/user

This XML file does not appear to have any style information associated with it. The

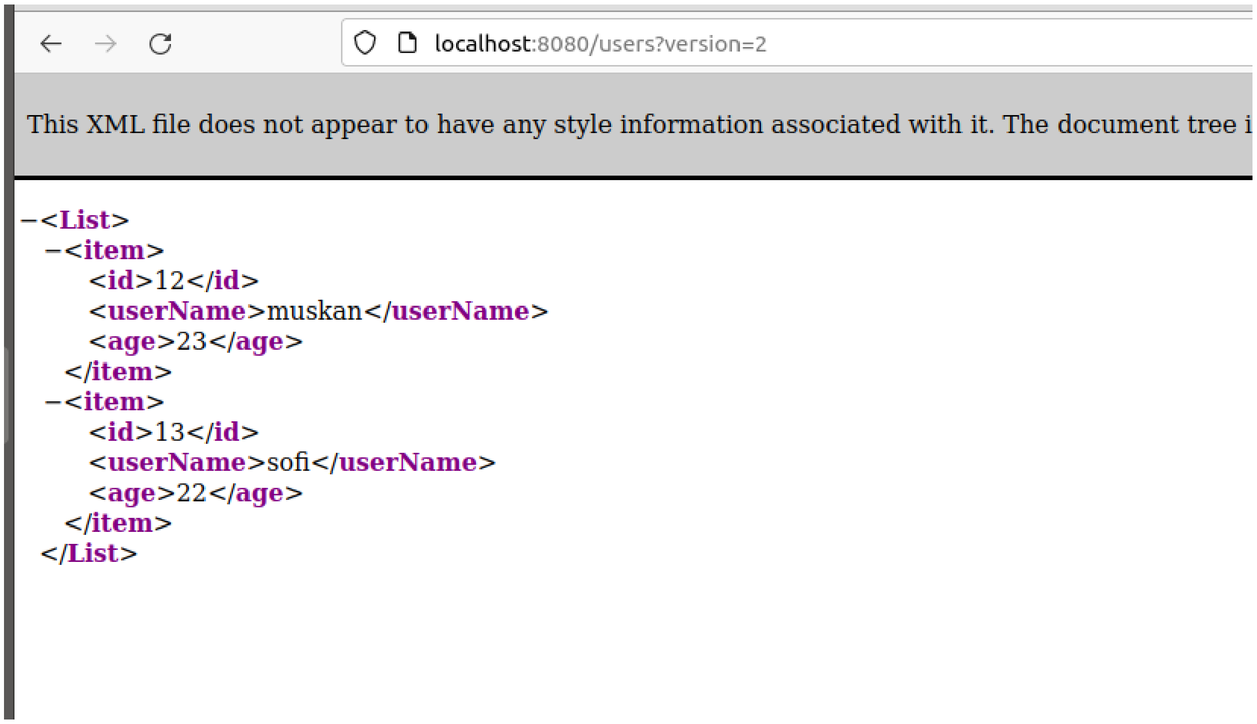
```
-<UserV2>
   <firstName>Muskan</firstName>
   <lastName>lama</lastName>
 </UserV2>
```


localhost:8080/v1/user

This XML file does not appear to have any style information associated wi

```
-<UserV1>
   <name>muskan</name>
 </UserV1>
```

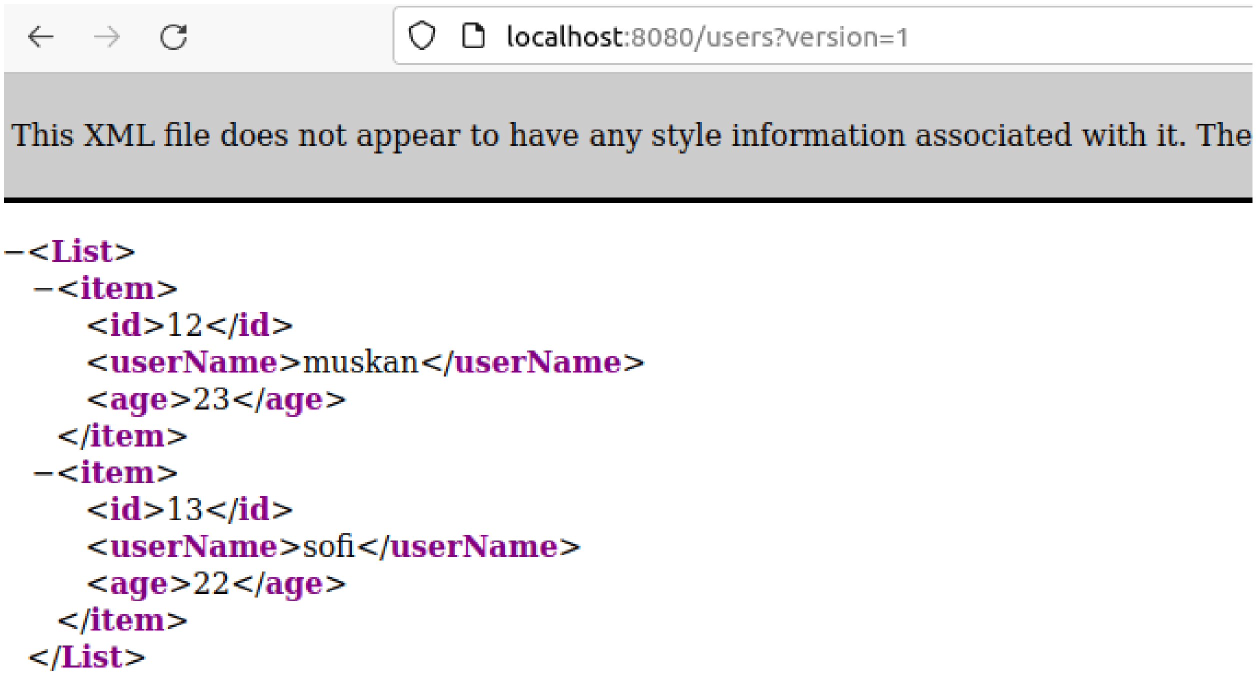
Request Parameter versioning



localhost:8080/users?version=2

This XML file does not appear to have any style information associated with it. The document tree i

```
–<List>
  –<item>
      <id>12</id>
      <userName>muskan</userName>
      <age>23</age>
    </item>
  –<item>
      <id>13</id>
      <userName>sofi</userName>
      <age>22</age>
    </item>
  </List>
```

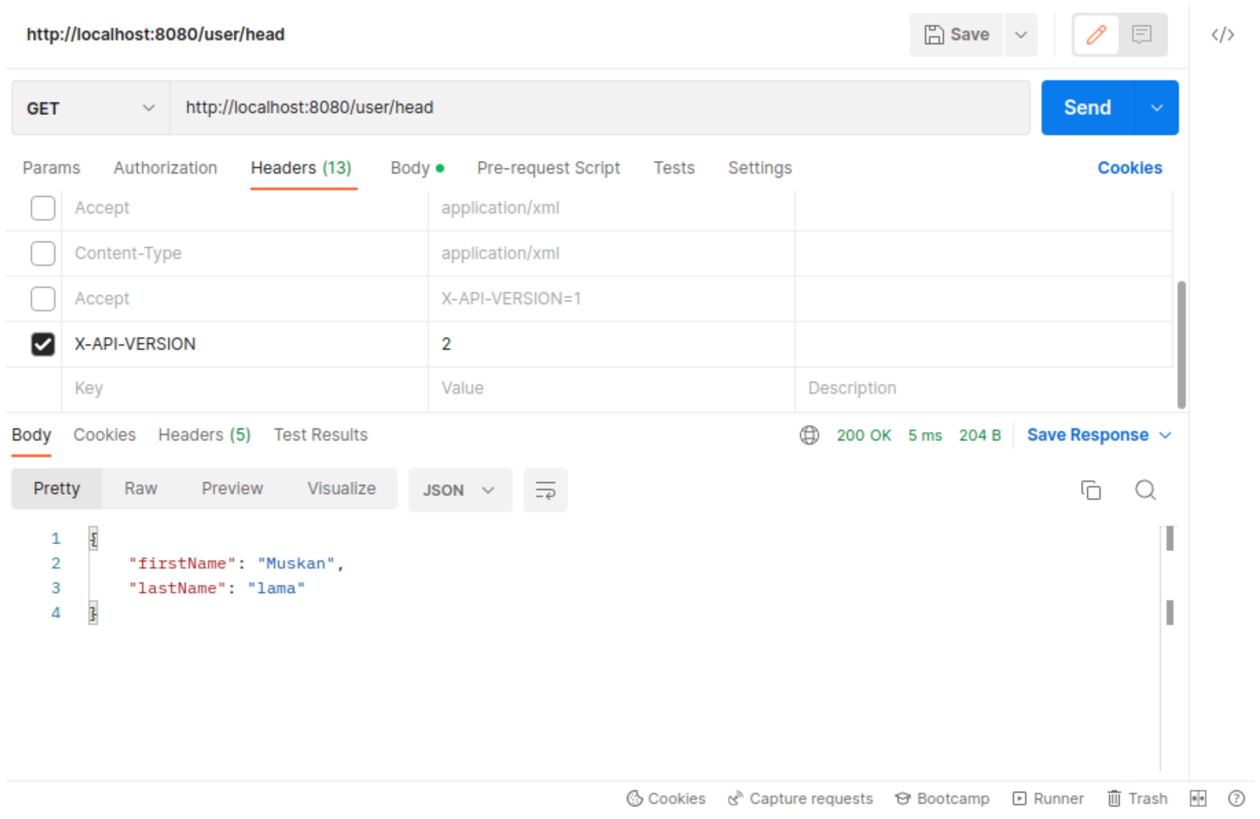


localhost:8080/users?version=1

This XML file does not appear to have any style information associated with it. The

```
–<List>
  –<item>
      <id>12</id>
      <userName>muskan</userName>
      <age>23</age>
    </item>
  –<item>
      <id>13</id>
      <userName>sofi</userName>
      <age>22</age>
    </item>
  </List>
```

Custom Header Versioning

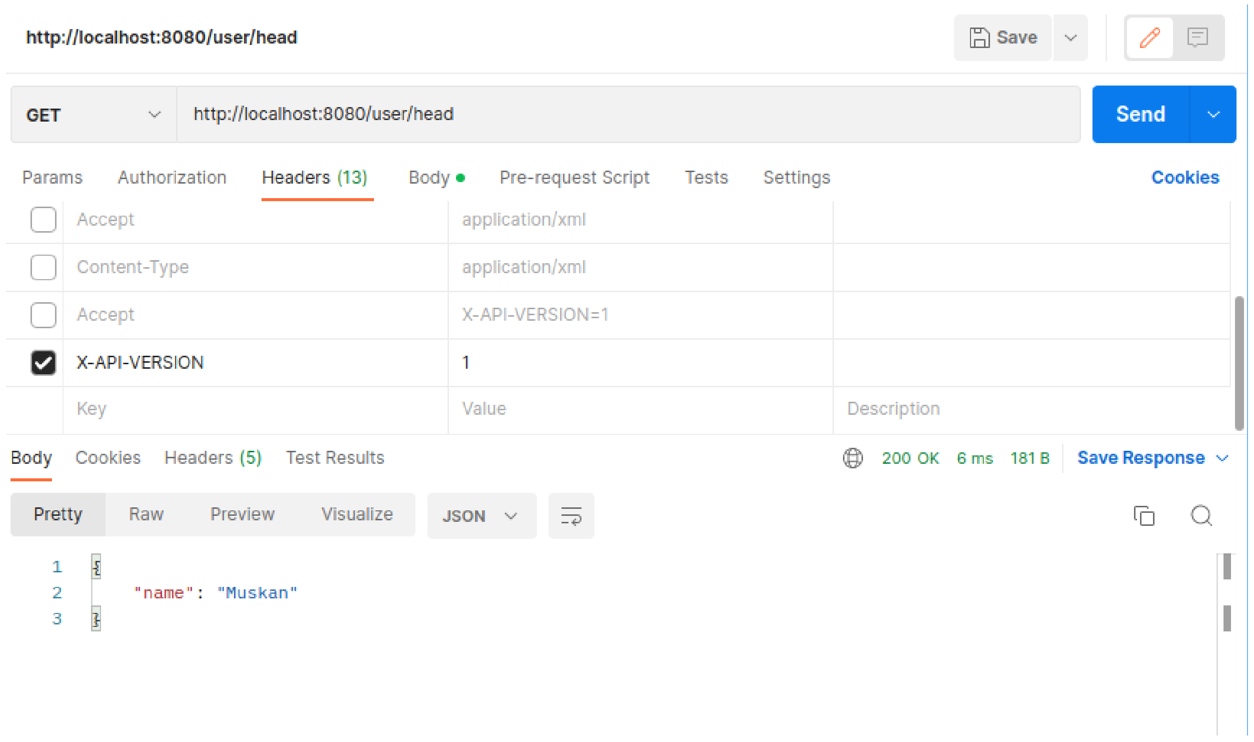

http://localhost:8080/user/head

GET http://localhost:8080/user/head

	Key	Value	Description
☐	Accept	application/xml	
☐	Content-Type	application/xml	
☐	Accept	X-API-VERSION=1	
☑	X-API-VERSION	2	

200 OK 5 ms 204 B

```
{
    "firstName": "Muskan",
    "lastName": "lama"
}
```


http://localhost:8080/user/head

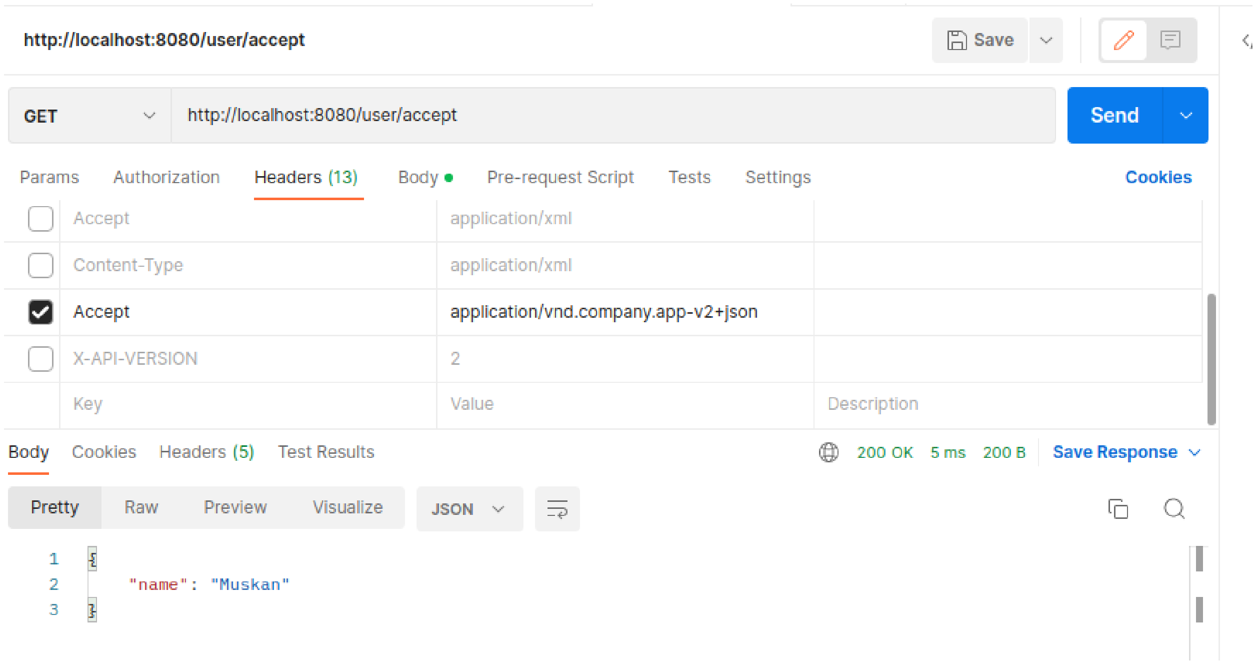
GET http://localhost:8080/user/head

	Key	Value	Description
☐	Accept	application/xml	
☐	Content-Type	application/xml	
☐	Accept	X-API-VERSION=1	
☑	X-API-VERSION	1	

200 OK 6 ms 181 B

```
{
    "name": "Muskan"
}
```

MimeType Versioning



http://localhost:8080/user/accept

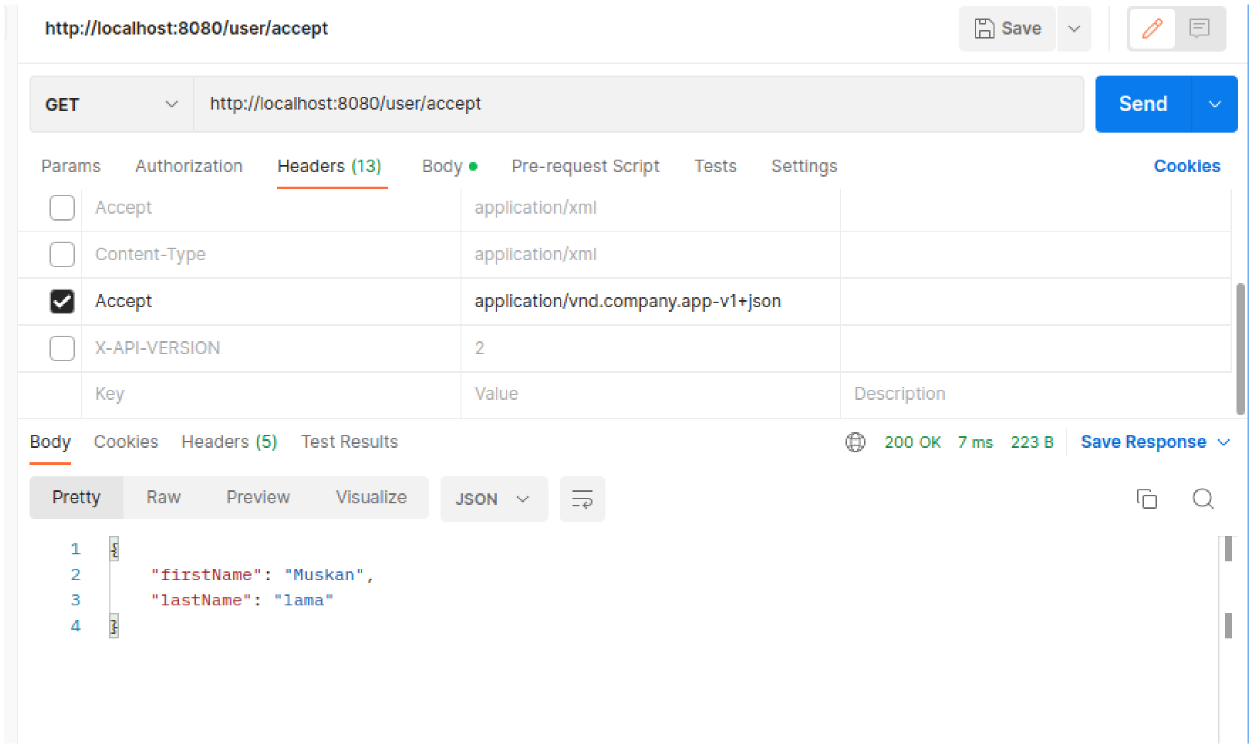
GET http://localhost:8080/user/accept

Headers (13)

	Key	Value	Description
☐	Accept	application/xml	
☐	Content-Type	application/xml	
☑	Accept	application/vnd.company.app-v2+json	
☐	X-API-VERSION	2	

200 OK 5 ms 200 B

```
{
    "name": "Muskan"
}
```



http://localhost:8080/user/accept

GET http://localhost:8080/user/accept

Headers (13)

	Key	Value	Description
☐	Accept	application/xml	
☐	Content-Type	application/xml	
☑	Accept	application/vnd.company.app-v1+json	
☐	X-API-VERSION	2	

200 OK 7 ms 223 B

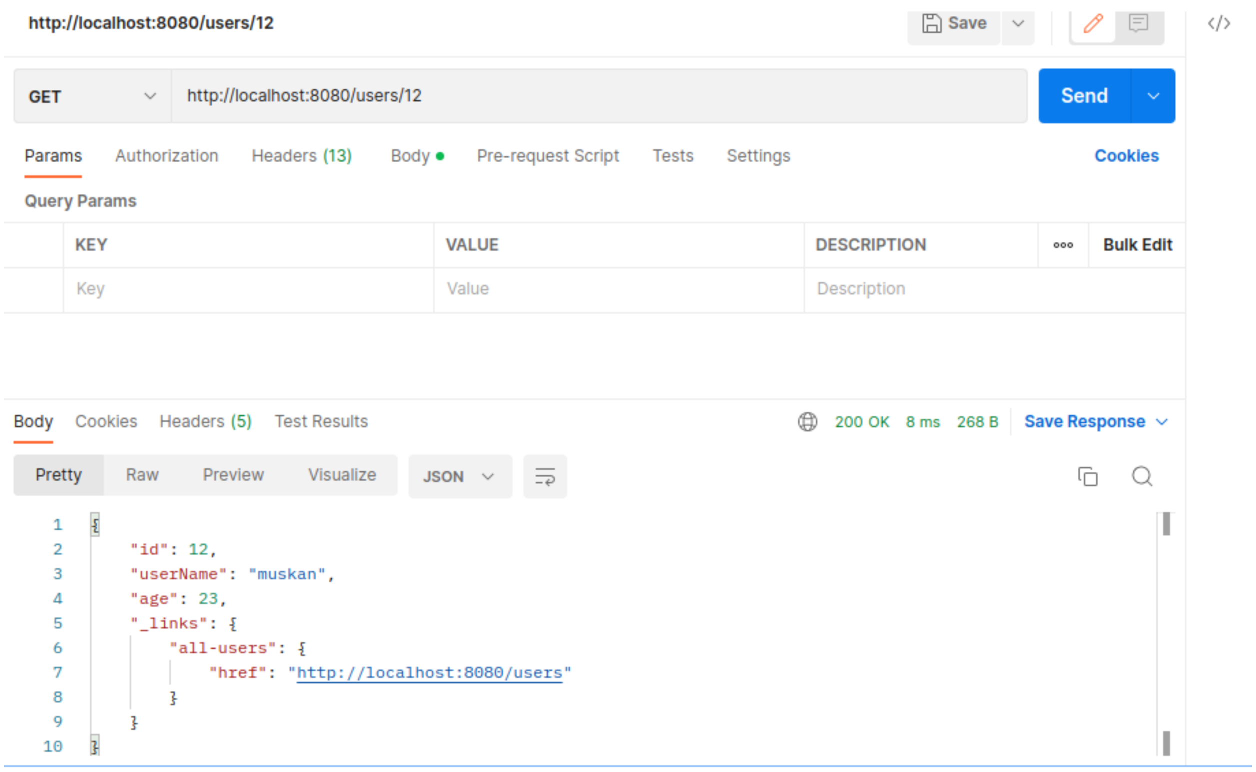
```
{
    "firstName": "Muskan",
    "lastName": "lama"
}
```

***HATEOAS**

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

```
//getting Pathvariable
//Implement GET
// http request using path variable top get one employee
//HATEOS
//content negotiation
@GetMapping("/users/{id}")
public EntityModel<Employee> retrieveUser(@PathVariable int id) throws
IdNotFoundException {
   Employee employee = service.findOne(id);

   //throwing exception when id is not found
   if(employee==null)
       throw new IdNotFoundException("id:"+id);


   //using entity model
   // represents RepresentationModel containing only single entity and related
links
   EntityModel<Employee> entityModel = EntityModel.of(employee);
   //we can use WebMvcLinkBuilder to create links pointing to controller
classes and it's methods.
   WebMvcLinkBuilder link =  linkTo(methodOn(this.getClass()).getAll());
 // adding link
   entityModel.add(link.withRel("all-users"));

   return entityModel;
}
```

Output



GET http://localhost:8080/users/12

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
    "id": 12,
    "userName": "muskan",
    "age": 23,
    "_links": {
        "all-users": {
            "href": "http://localhost:8080/users"
        }
    }
}
```