RestFul Web Service Part 2

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

File=HelloWorldController.java

```
package com.example.RestfulWebService.helloWorld;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import java.util.Locale;

@RestController
public class HelloWorldController {

  @Autowired
  public MessageSource messageSource;


  //Ques1.....................................................
  @GetMapping(path = "/hello-world-internationalised")
  public String helloWorldInter(@RequestHeader(name = "Accept-Language", required = false) Locale locale) {
     return messageSource.getMessage("good.morning.message", null, locale);
  }
}
```

---

file=messages.properties

```
#//this will contain the default ones -Good Morning is default one

good.morning.message=Good Morning
```

---

file=messages_gr.properties

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

---

file=messages_sw.properties

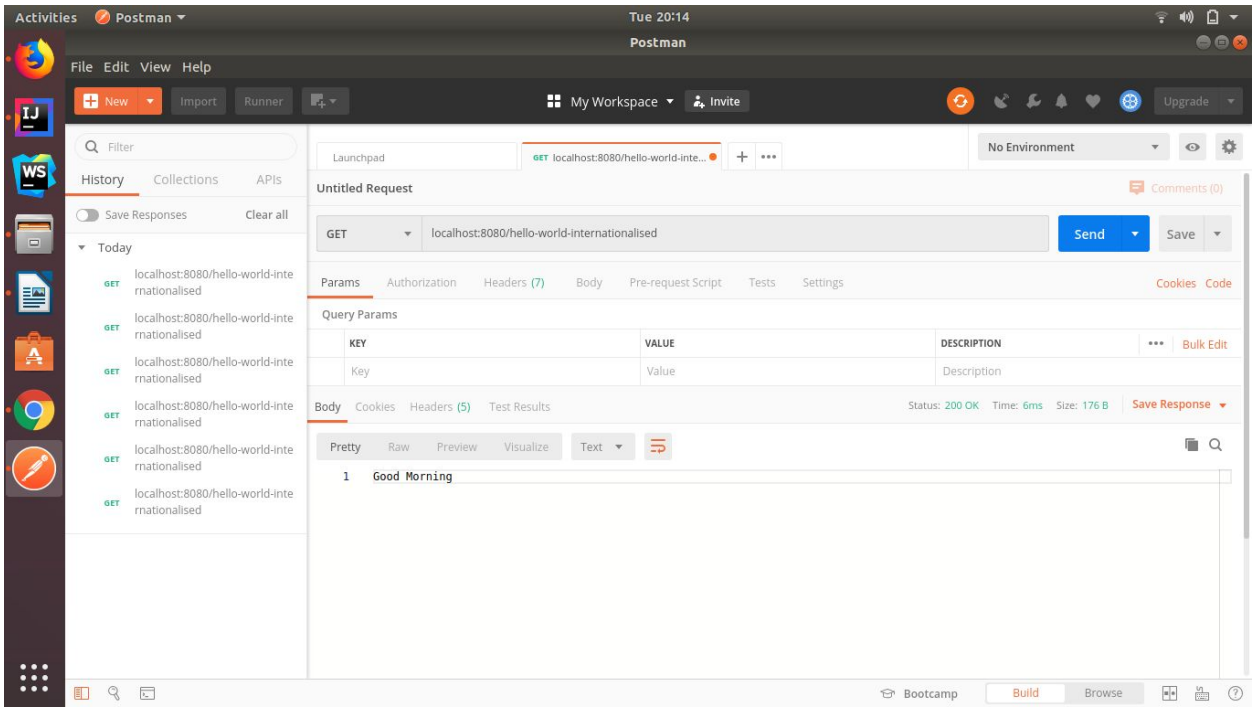
```
good.morning.message=God morgon
```

---

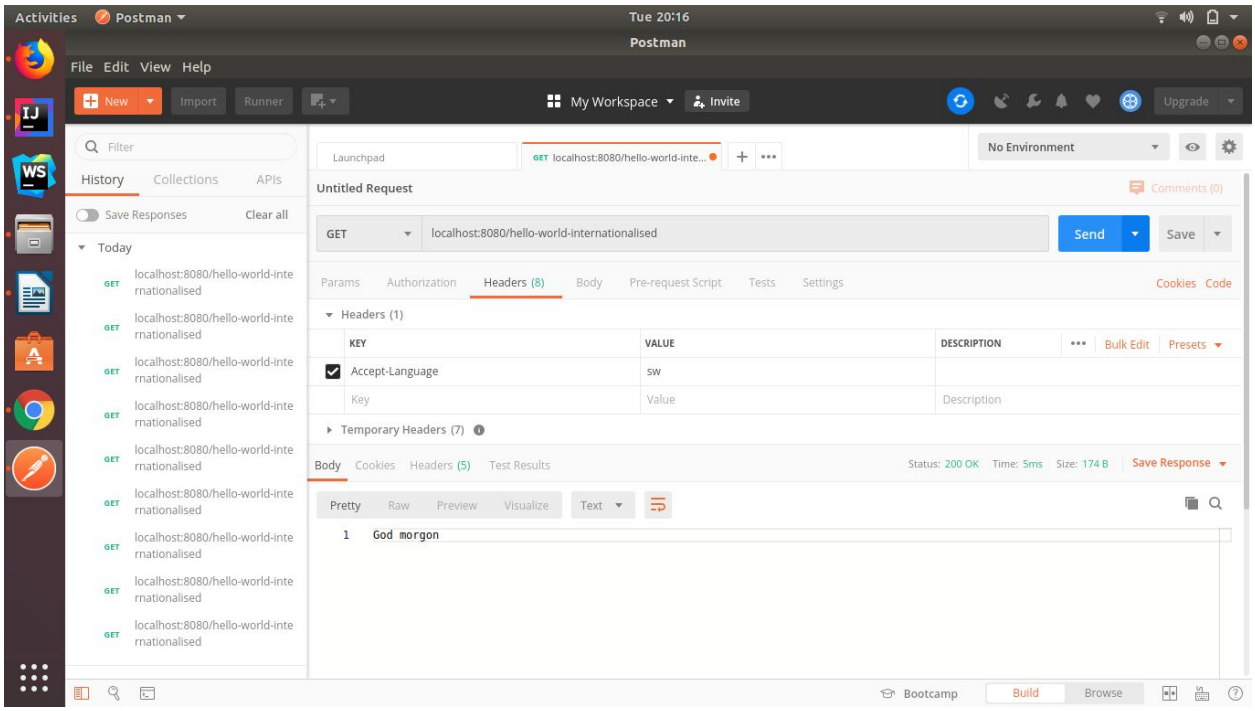
file=RestFulWebServiceApplication

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.context.support.ResourceBundleMessageSource;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import java.util.Locale;

@SpringBootApplication
public class RestfulWebServiceApplication {

  public static void main(String[] args) {
    SpringApplication.run(RestfulWebServiceApplication.class, args);
  }


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


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

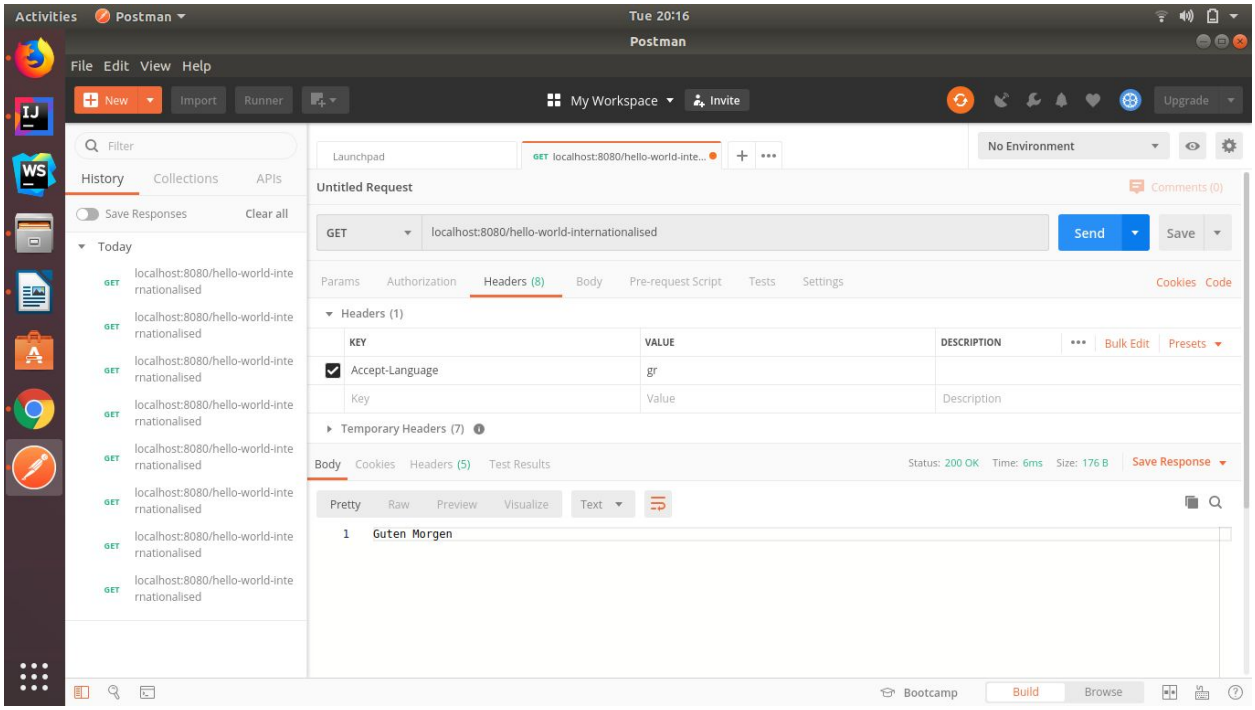
OUTPUT- messages is shown in English, keeping English as default.



Message is German.



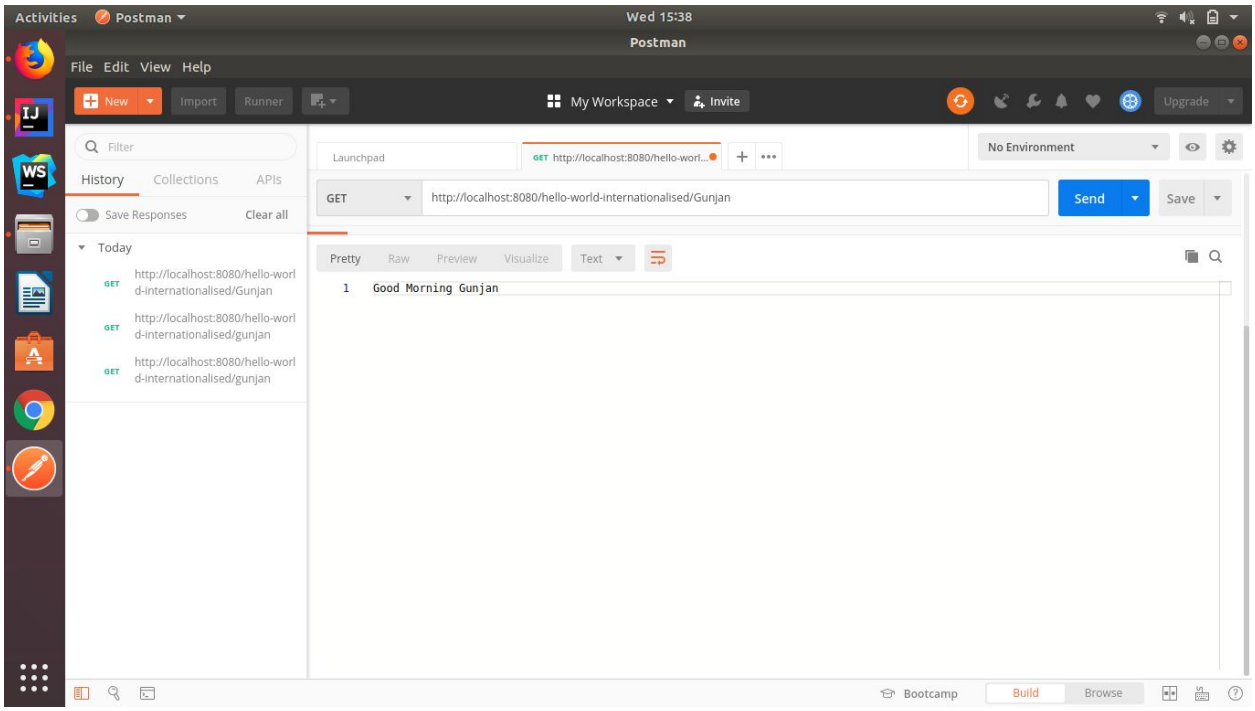
Message is shown in Swedish



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

```
package com.example.RestfulWebService.helloWorld;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

import java.util.Locale;

@RestController
public class HelloWorldController {

  @Autowired
  public MessageSource messageSource;


  //Ques2.....
  //Using path parameter
```

```
  @GetMapping(path = "/hello-world-internationalised/{name}")
  public String helloWorldInter(@PathVariable String name, @RequestHeader(name = "Accept-Language",
required    = false) Locale locale) {
   return messageSource.getMessage("good.morning.message", null, locale)/*+String.format(" %s
",name)*/+name;
}
}
```



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

File=build.gradle

```
plugins {
  id 'org.springframework.boot' version '2.2.5.RELEASE'
  id 'io.spring.dependency-management' version '1.0.9.RELEASE'
  id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
  mavenCentral()
}

dependencies {

  implementation 'org.springframework.boot:spring-boot-starter-web'
  //Ques 3.....
  compile 'com.fasterxml.jackson.dataformat:jackson-dataformat-xml'
  compile group: 'org.springframework.boot', name: 'spring-boot-starter-actuator'
  testImplementation('org.springframework.boot:spring-boot-starter-test') {
     exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
  }
}
```

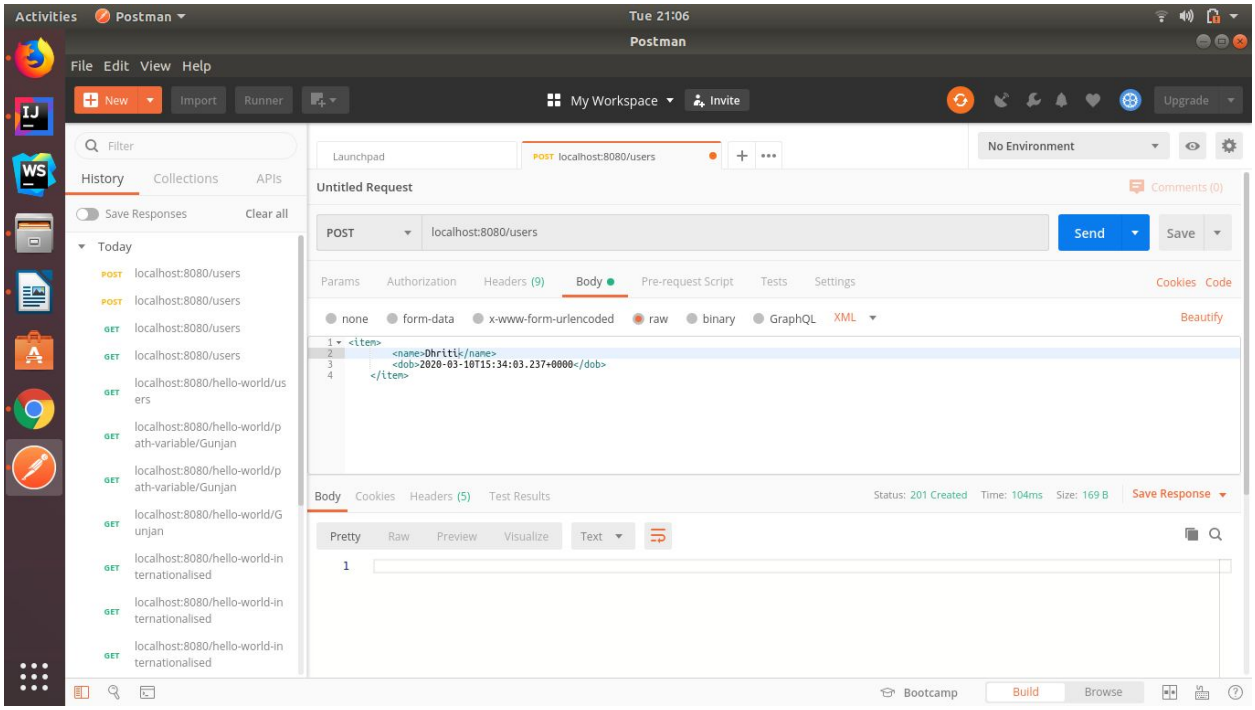
```
test {
  useJUnitPlatform()
}
```

File=userresource.java

```
package com.example.RestfulWebService.user;
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 javax.validation.Valid;
import java.net.URI;
import java.util.List;



@RestController
public class userresource {

  @Autowired
  private Userdoa service;

  @GetMapping("/users")
  public List<User> rettriveAllUSers(){
     return service.findAll();
  }

  @GetMapping("/users/{id}")
  public User rettriveUSer(@PathVariable  int id){
     User user=service.findOne(id);
     if(user==null)
        throw new UserNotFoundException("id-"+id);
     return user;
  }



   //Ques3 Create POST Method to create user details which can accept XML for user creation.
  @PostMapping("/users")
  public ResponseEntity<Object> createUser(@Valid @RequestBody User user){
        User savedUser=service.save(user);

     URI location=ServletUriComponentsBuilder
           .fromCurrentRequest()
           .path("/{id}")
           .buildAndExpand(savedUser.getId()).toUri();
```

```
        return ResponseEntity.created(location).build();
    }

}
```

OUTPUT-

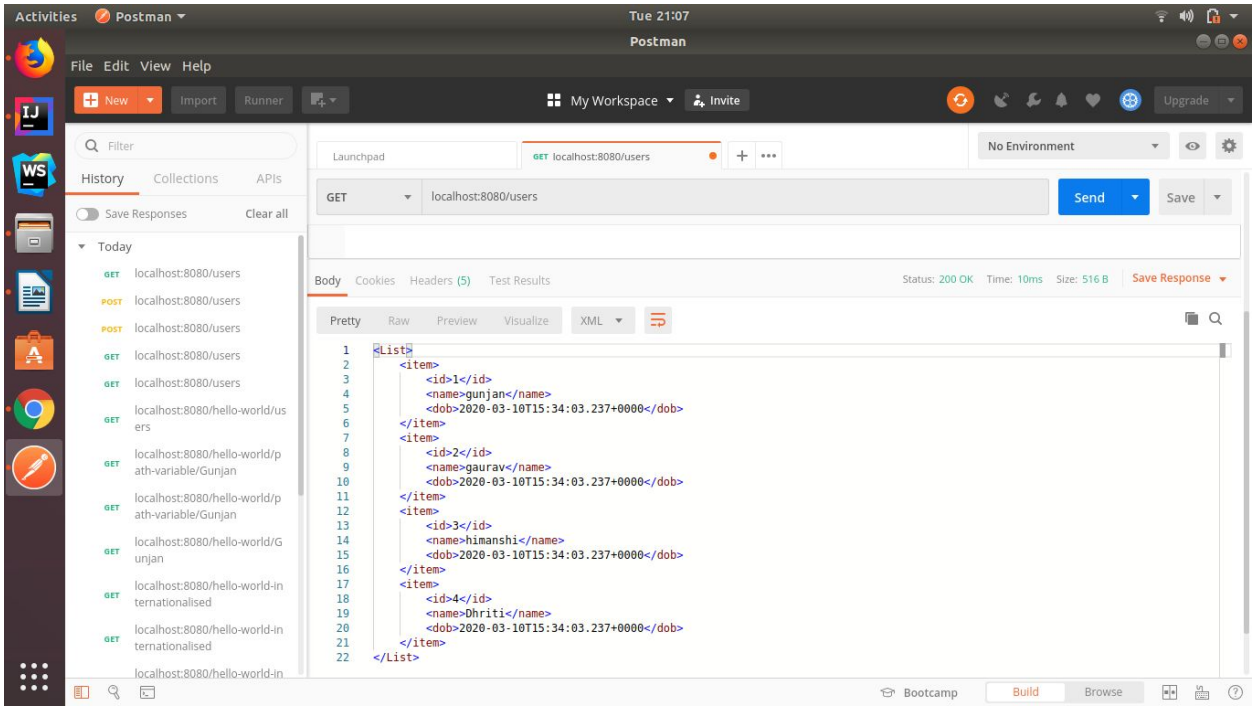


4. Create GET Method to fetch the list of users in XML format.

File=userresource.java

```
package com.example.RestfulWebService.user;
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 javax.validation.Valid;
import java.net.URI;
import java.util.List;



@RestController
public class userresource {

  @Autowired
  private Userdoa service;

   //Ques4 Create GET Method to fetch the list of users in XML format.
  @GetMapping("/users")
  public List<User> rettriveAllUSers(){
     return service.findAll();
  }

  @GetMapping("/users/{id}")
  public User rettriveUSer(@PathVariable  int id){
     User user=service.findOne(id);
```

```
    if(user==null)
        throw new UserNotFoundException("id-"+id);
    return user;
  }

  @PostMapping("/users")
  public ResponseEntity<Object> createUser(@Valid @RequestBody User user){
        User savedUser=service.save(user);

    URI location=ServletUriComponentsBuilder
          .fromCurrentRequest()
          .path("/{id}")
          .buildAndExpand(savedUser.getId()).toUri();
     return ResponseEntity.created(location).build();
  }

}
```



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

File=build.gradle

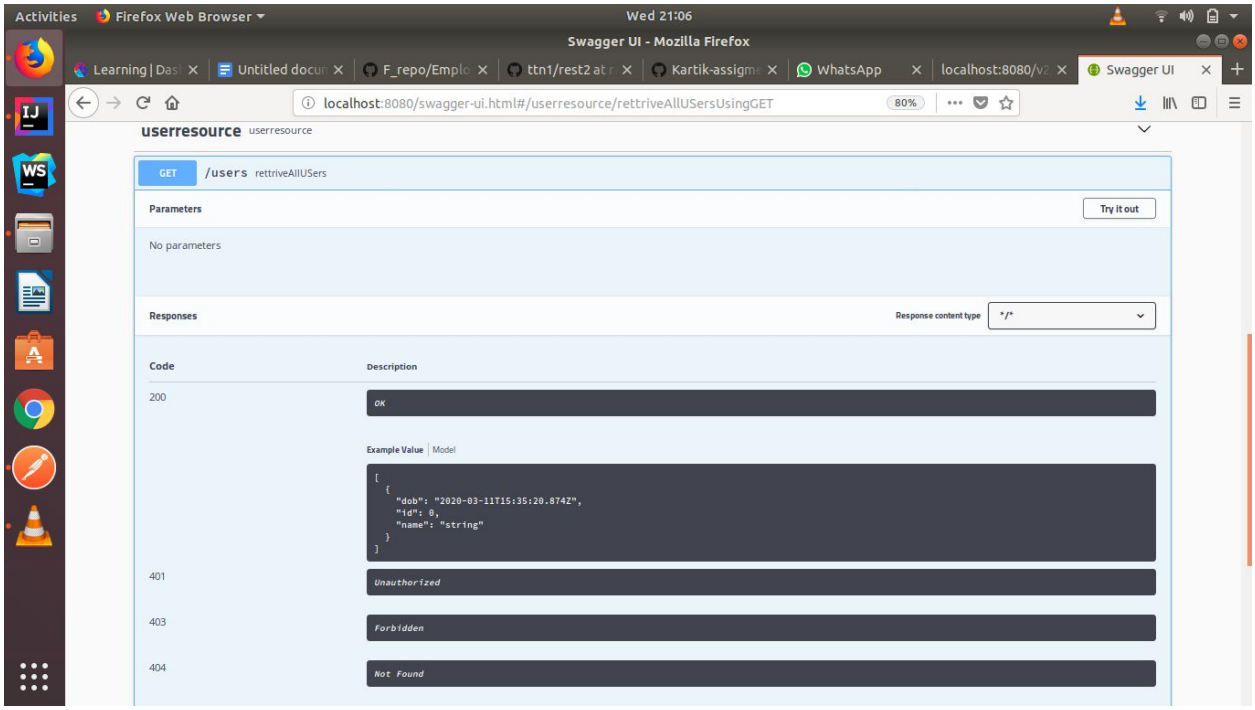
```
plugins {
  id 'org.springframework.boot' version '2.2.5.RELEASE'
  id 'io.spring.dependency-management' version '1.0.9.RELEASE'
  id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
  mavenCentral()
}

dependencies {

  implementation 'org.springframework.boot:spring-boot-starter-web'
  compile 'com.fasterxml.jackson.dataformat:jackson-dataformat-xml'
  compile group: 'org.springframework.boot', name: 'spring-boot-starter-actuator'
  //Ques5 Configure Swagger Plugin..
  compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.9.2'
  compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.9.2'

  testImplementation('org.springframework.boot:spring-boot-starter-test') {
     exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
  }
}
```

```
test {
  useJUnitPlatform()
}
```

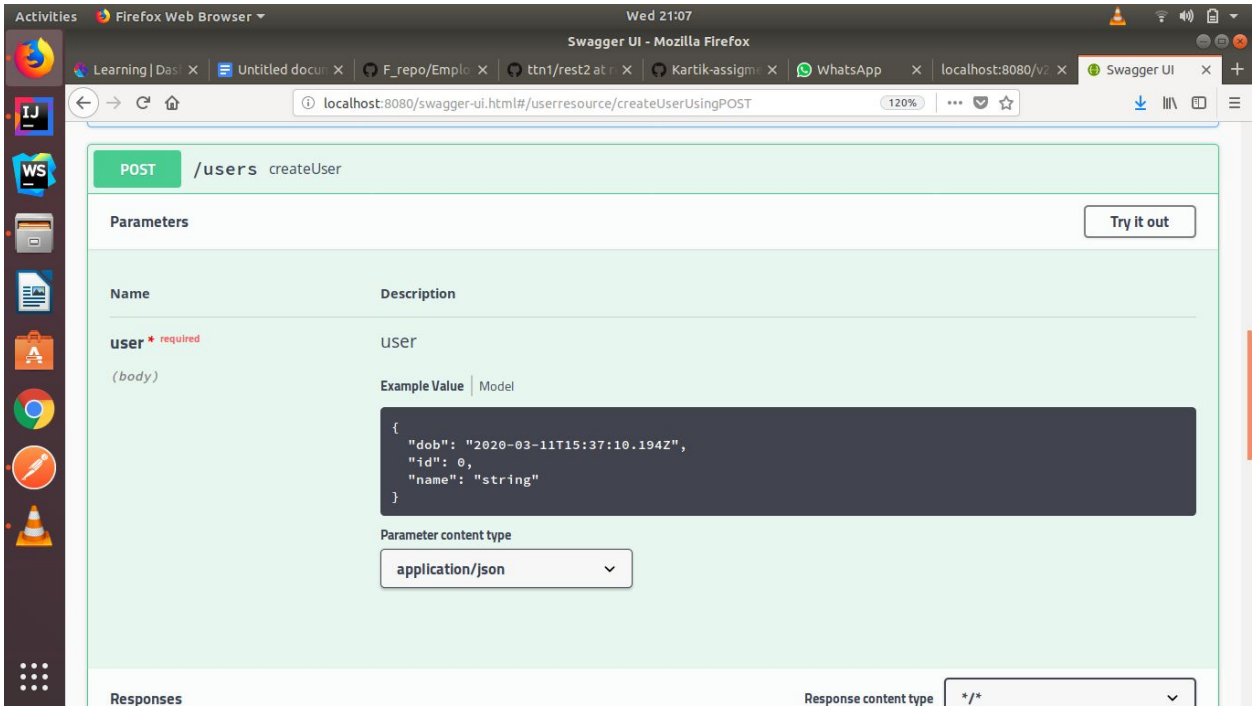
File=SwaggerConfig.java

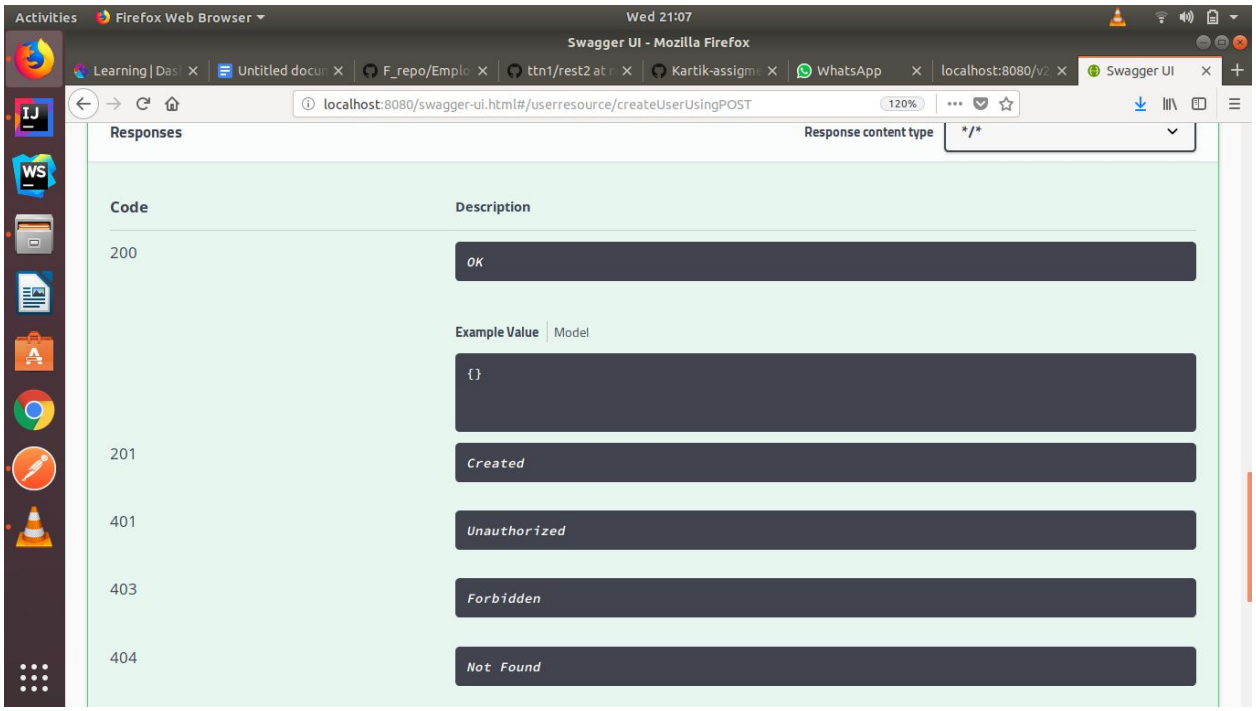
```
package com.example.RestfulWebService;

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


@Configuration
@EnableSwagger2
public class SwaggerConfig {

   //Bean-Docket(in this i want to tell i will use swagger2
   //Docket is already defined in springfox-swagger2 documentation
   @Bean
   public Docket api(){
      return new Docket(DocumentationType.SWAGGER_2);
   }
}
```

i) Get details of User using GET request.

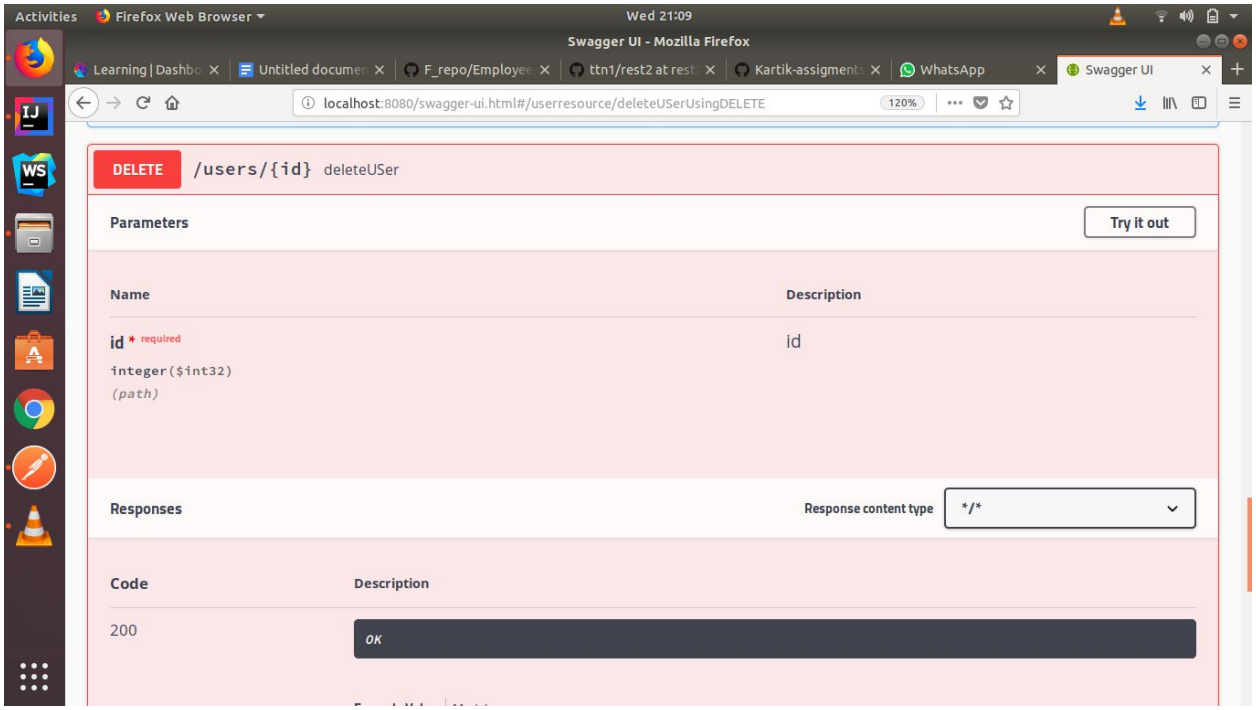


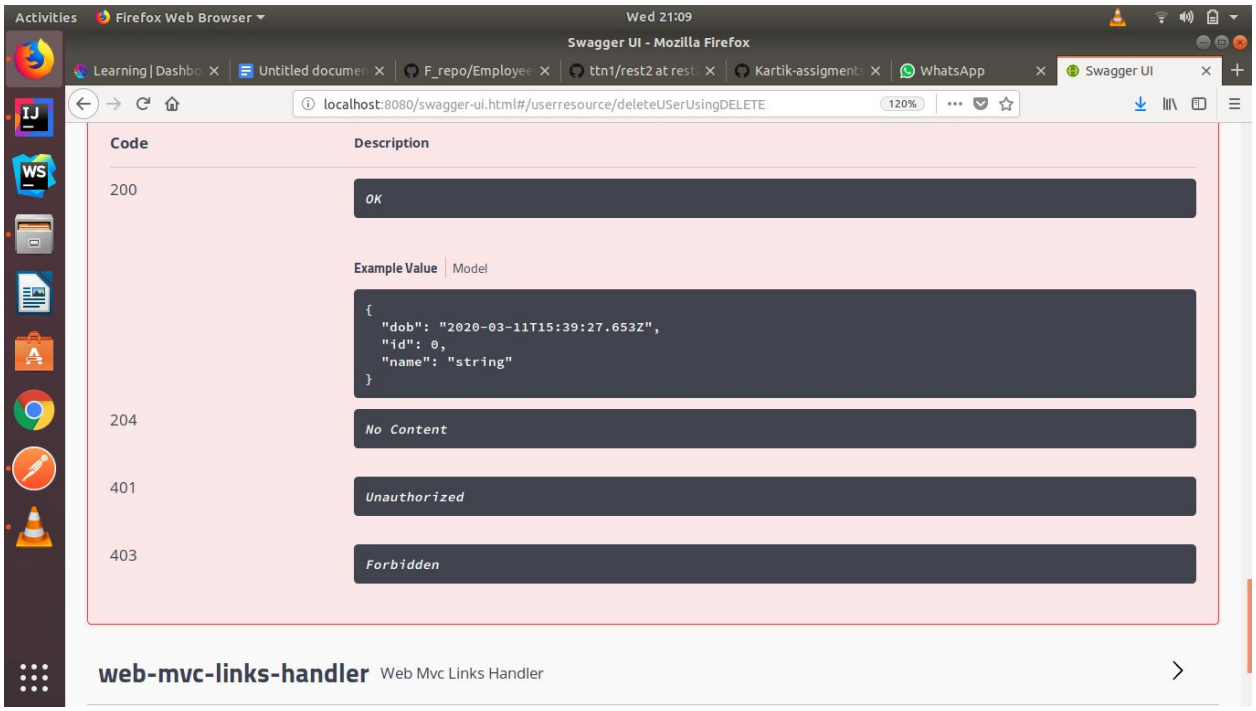
ii) Save details of the user using POST request.





iii) Delete a user using DELETE request.


Activities
Firefox Web Browser
Wed 21:09
Swagger UI - Mozilla Firefox
Swagger UI
localhost:8080/swagger-ui.html#/userresource/deleteUSerUsingDELETE
DELETE
/users/{id}
deleteUSer
Parameters
Try it out
Name
Description
id * required
integer($int32)
(path)
id
Responses
Response content type
/
Code
Description
200
OK


Activities
Firefox Web Browser
Wed 21:09
Swagger UI - Mozilla Firefox
Swagger UI
localhost:8080/swagger-ui.html#/userresource/deleteUSerUsingDELETE
Code
Description
200
OK
Example Value
Model
{
"dob": "2020-03-11T15:39:27.653Z",
"id": 0,
"name": "string"
}
204
No Content
401
Unauthorized
403
Forbidden
web-mvc-links-handler
Web Mvc Links Handler

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.

Using @ApiModel

File=Userresource.java

```
package com.example.RestfulWebService.user;
import io.swagger.annotations.ApiModelProperty;
import org.springframework.beans.factory.annotation.Autowired;
//import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
//import org.springframework.hateoas.EntityModel;
import javax.validation.Valid;
import java.net.URI;
import java.util.List;

//import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;
//import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.methodOn;

@RestController
public class userresource {

  @Autowired
  private Userdoa service;

  //GET /users
  //retrieveAllUsers
  //Ques4 Create GET Method to fetch the list of users in XML format.
  @ApiModelProperty(name = "RetrieveAll")
  @GetMapping("/users")
  public List<User> rettriveAllUSers(){
     return service.findAll();
  }


  //GET /users/id
  //retrieveParticularUsers
  //this should return the status code 200 on success
  @ApiModelProperty(name = "RetrieveParticular")
  @GetMapping("/users/{id}")
  public User rettriveUSer(@PathVariable  int id){
     User user=service.findOne(id);
     if(user==null)
        throw new UserNotFoundException("id-"+id);
     return user;
  }
```

```
  //input-details of user
  //Output-Created(status) and return the created uri
  //Ques3 Create POST Method to create user details which can accept XML for user creation.
  @ApiModelProperty(name = "Create New User")
  @PostMapping("/users")
  public ResponseEntity<Object> createUser(@Valid @RequestBody User user){
        User savedUser=service.save(user);

        //CREATED
        //user/4-to show this is posted
        //user/{id}    user/savedUser.getId()
     //making created http 201 and loation of the newly created uri

     URI location=ServletUriComponentsBuilder
           .fromCurrentRequest()
           .path("/{id}")
           .buildAndExpand(savedUser.getId()).toUri(); //for finding correct uri and appending id to it
      return ResponseEntity.created(location).build(); //to create correct http status
  }


  @ApiModelProperty(name="Deleting a prticular User")
  @DeleteMapping("/users/{id}")
  public User deleteUSer(@PathVariable  int id){
     User user=service.deleteByID(id);
     if(user==null)
        throw new UserNotFoundException("id-"+id);
     return user;
  }
}
```

File=SwaggerConfig.java

```
package com.example.RestfulWebService;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.service.VendorExtension;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;


@Configuration
@EnableSwagger2
public class SwaggerConfig {
```

```
  public static final Contact DEFAULT_CONTACT = new Contact("Gunjan", "tothenew.com",
"gunjan.dawar@gmail.com");
  public static final ApiInfo DEFAULT_API_INFO = new ApiInfo("Api Documentation", "Api Documentation",
"1.0", "urn:tos",
        DEFAULT_CONTACT, "Apache 2.0", "http://www.apache.org/licenses/LICENSE-2.0", new
ArrayList<VendorExtension>());


  //Bean-Docket(in this i want to tell i will use swagger2
  //Docket is already defined in springfox-swagger2 documentation
  @Bean
  public Docket api(){
    return new Docket(DocumentationType.SWAGGER_2)
        .apiInfo(DEFAULT_API_INFO);
  }
}
```

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

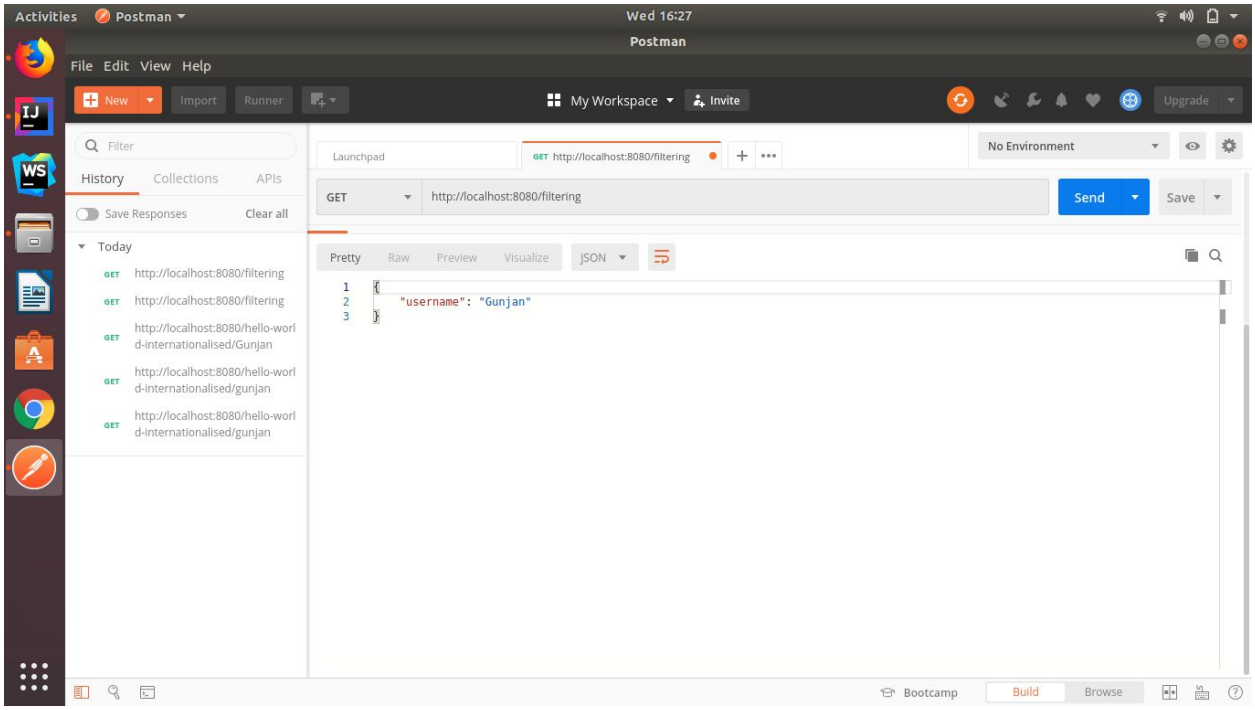
File=UserDetails.java

```
package com.example.RestfulWebService.Filtering;

import com.fasterxml.jackson.annotation.JsonIgnore;

public class UserDetails {

  private String username;
  @JsonIgnore
  private String password;

  public String getUsername() {
    return username;
  }

  public void setUsername(String username) {
    this.username = username;
  }

  public String getPassword() {
    return password;
  }

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

  public UserDetails(String username, String password) {
    this.username = username;
    this.password = password;
  }
}
```

File=Filteringcontroller.java

```
package com.example.RestfulWebService.Filtering;

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

//Ques8. Create API which saves details of User (along with the password) but on successfully saving returns only non-critical data. (Use static filtering)
@RestController
public class FilteringController {
```

```
  @GetMapping("/filtering")
  public UserDetails retriveUserDetails(){
     return new UserDetails("Gunjan","12345678");
  }
}
```



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

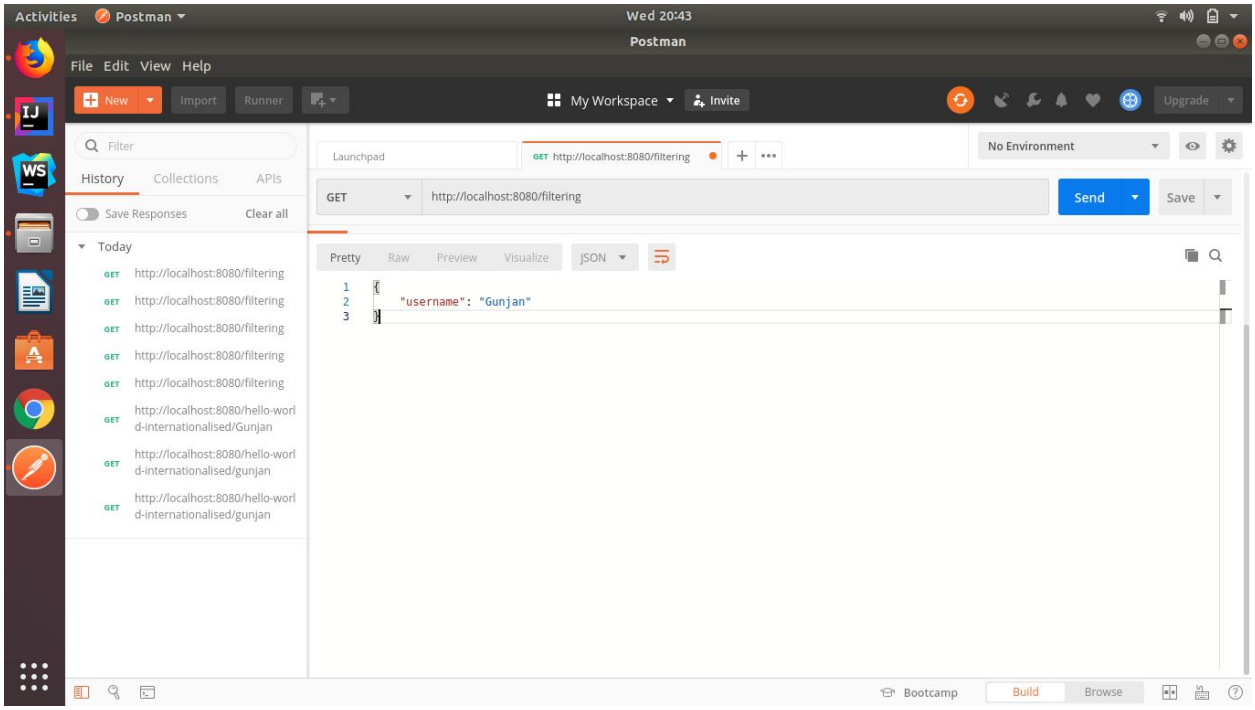
File=UserDetails.java

```
 package com.example.RestfulWebService.Filtering;


import com.fasterxml.jackson.annotation.JsonFilter;
import com.fasterxml.jackson.annotation.JsonIgnore;

@JsonFilter("UserDetailsFilter")
public class UserDetails {

  private String username;
  @JsonIgnore
  private String password;

  public String getUsername() {
    return username;
  }

  public void setUsername(String username) {
    this.username = username;
  }

  public String getPassword() {
    return password;
  }

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

  public UserDetails(String username, String password) {
    this.username = username;
    this.password = password;
  }
}
```

File=FilteringController.java

```
package com.example.RestfulWebService.Filtering;

import com.fasterxml.jackson.databind.ser.FilterProvider;
import com.fasterxml.jackson.databind.ser.impl.SimpleBeanPropertyFilter;
import com.fasterxml.jackson.databind.ser.impl.SimpleFilterProvider;
```

```java
import org.springframework.http.converter.cbor.MappingJackson2CborHttpMessageConverter;
import org.springframework.http.converter.json.MappingJacksonValue;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

//Ques9. Create another API that does the same by using Dynamic Filtering.
@RestController
public class FilteringController {

  @GetMapping("/filtering")
  public MappingJacksonValue retriveUserDetails(){

    UserDetails userDetails= new UserDetails("Gunjan","12345678");
    SimpleBeanPropertyFilter filter=SimpleBeanPropertyFilter.
        filterOutAllExcept("username");
    FilterProvider filters=new SimpleFilterProvider().addFilter("UserDetailsFilter",filter);

    MappingJacksonValue mapping=new MappingJacksonValue(userDetails);
    mapping.setFilters(filters);

    return mapping;

  }

}
```


Activities
Postman
Wed 20:43
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/hello-world-internationalised/Gunjan
GET http://localhost:8080/hello-world-internationalised/gunjan
GET http://localhost:8080/hello-world-internationalised/gunjan
Launchpad
GET http://localhost:8080/filtering
No Environment
GET
http://localhost:8080/filtering
Send
Save
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "username": "Gunjan"
3 }
Bootcamp
Build
Browse

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:

- URI versioning

File=UserV1.java

```
package com.example.RestfulWebService.Versioning;

public class UserV1 {
  public UserV1(){

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

  public String getName() {
     return name;
  }

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

  private String name;
}
```

File=UserV2.java

```
package com.example.RestfulWebService.Versioning;

public class UserV2 {
  public UserV2(){

  }
  public UserV2(Name name) {
     this.name = name;
  }
```

```
  private Name name;

  public Name getName() {
    return name;
  }

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

File=Name.java

```
package com.example.RestfulWebService.Versioning;

public class Name {
  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;
  }

  public Name(){
  }

  public Name(String firstname, String lastname) {
    this.firstname = firstname;
    this.lastname = lastname;
  }

  private String firstname;
  private String lastname;
}
```

---

*/*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

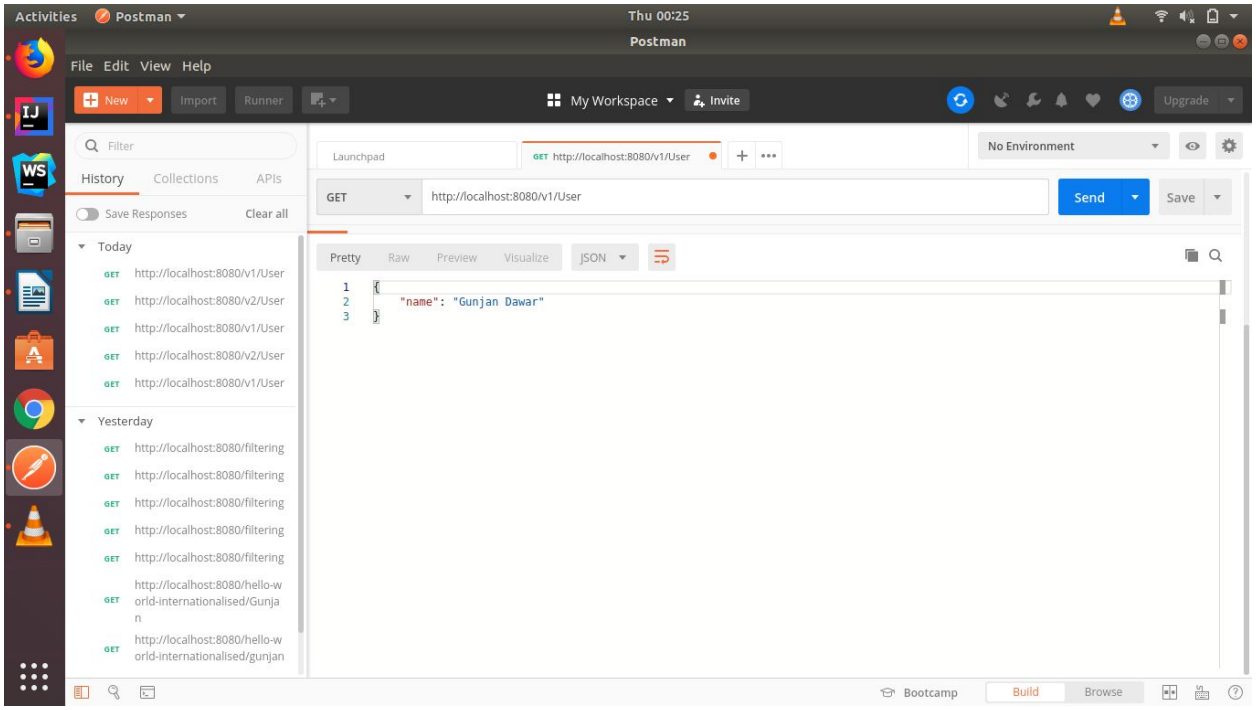
Request Parameter versioning

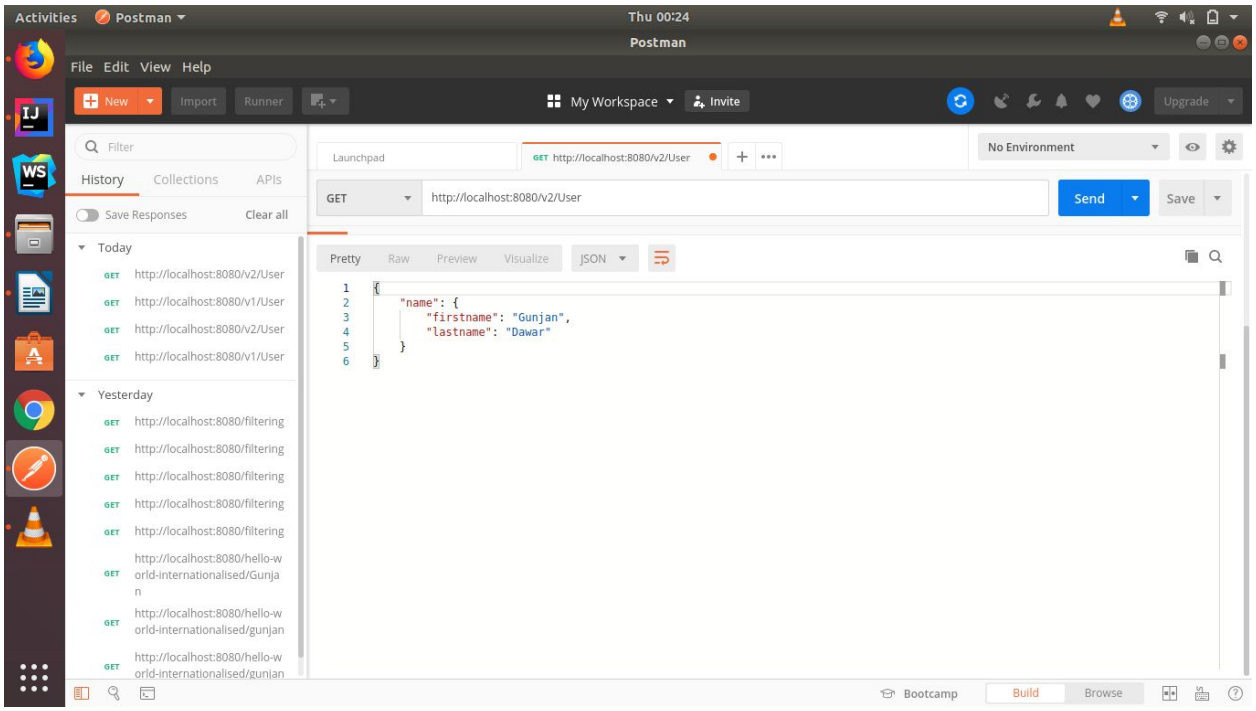
URI versioning

Custom Header Versioning/*

File=UserVersioningController.java

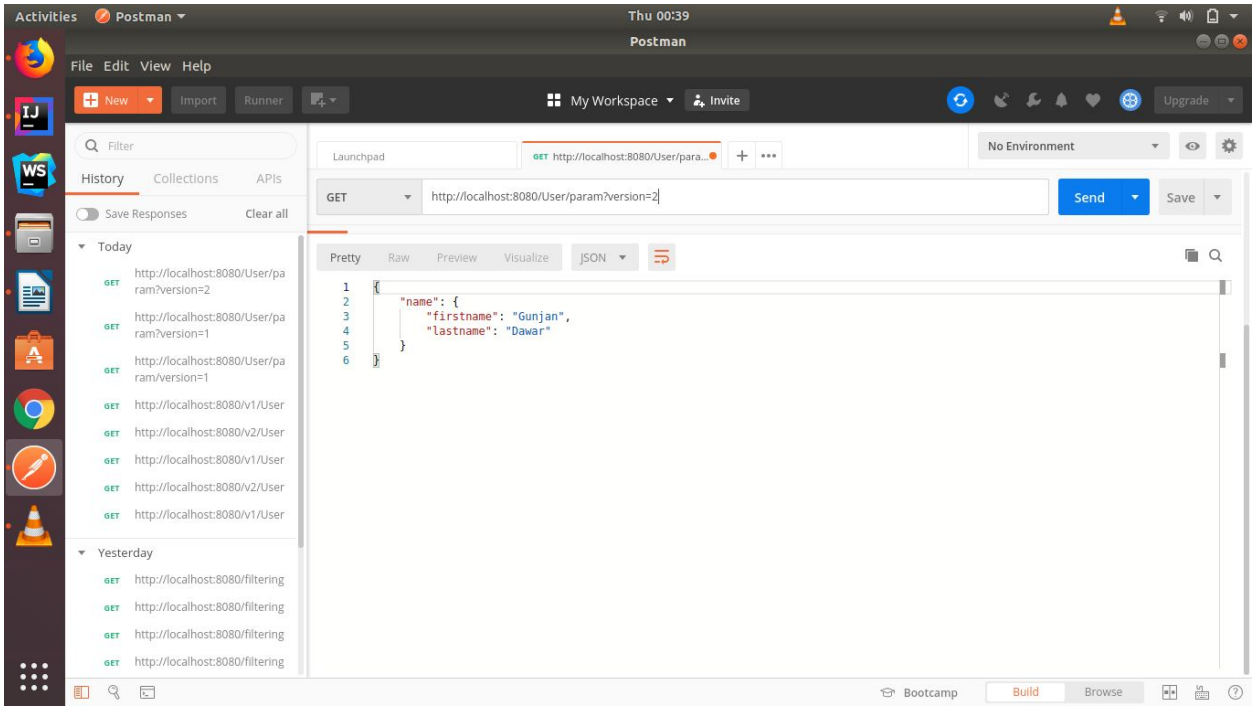
```
package com.example.RestfulWebService.Versioning;

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

@RestController
public class UserVersioningController {

  @GetMapping("v1/User")
  public UserV1 userV1(){
     return new UserV1("Gunjan Dawar");

  }
     @GetMapping("v2/User")
     public UserV2 userV2(){
        return new UserV2(new Name("Gunjan","Dawar"));
     }

}
```

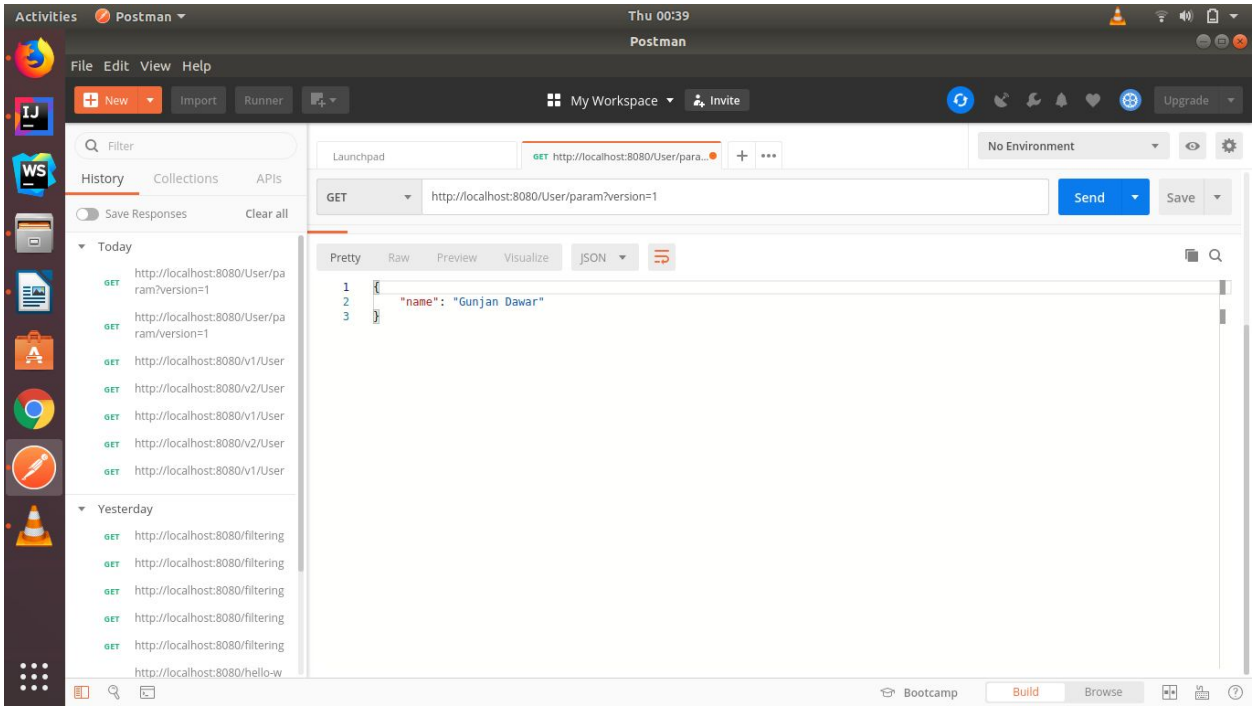

Activities
Postman
Thu 00:25
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
GET http://localhost:8080/v1/User
GET http://localhost:8080/v2/User
GET http://localhost:8080/v1/User
GET http://localhost:8080/v2/User
GET http://localhost:8080/v1/User
Yesterday
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/hello-world-internationalised/Gunjan
GET http://localhost:8080/hello-world-internationalised/gunjan
Launchpad
GET http://localhost:8080/v1/User
No Environment
GET
http://localhost:8080/v1/User
Send
Save
Pretty
Raw
Preview
Visualize
JSON
{
"name": "Gunjan Dawar"
}
Bootcamp
Build
Browse


Activities
Postman
Thu 00:24
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
GET http://localhost:8080/v2/User
GET http://localhost:8080/v1/User
GET http://localhost:8080/v2/User
GET http://localhost:8080/v1/User
Yesterday
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/filtering
GET http://localhost:8080/hello-world-internationalised/Gunjan
GET http://localhost:8080/hello-world-internationalised/gunjan
Launchpad
GET http://localhost:8080/v2/User
No Environment
GET
http://localhost:8080/v2/User
Send
Save
Pretty
Raw
Preview
Visualize
JSON
{
"name": {
"firstname": "Gunjan",
"lastname": "Dawar"
}
}
Bootcamp
Build
Browse

- Request Parameter versioning

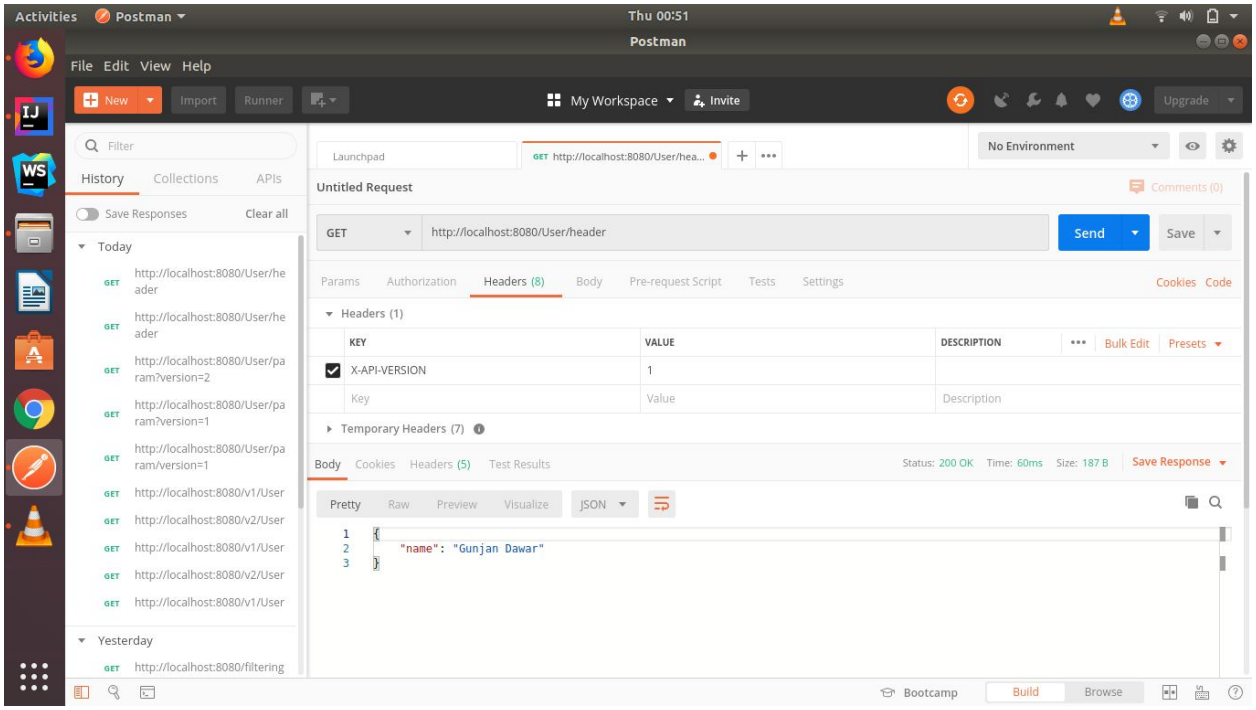
```java
package com.example.RestfulWebService.Versioning;

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

@RestController
public class UserVersioningController {


  //Request Parameter Versioning
  @GetMapping(value = "User/param",params = "version=1")
  public UserV1 paramV1(){
    return new UserV1("Gunjan Dawar");

  }


  @GetMapping(value = "User/param",params = "version=2")
  public UserV2 param2(){
    return new UserV2(new Name("Gunjan","Dawar"));
  }


}
```

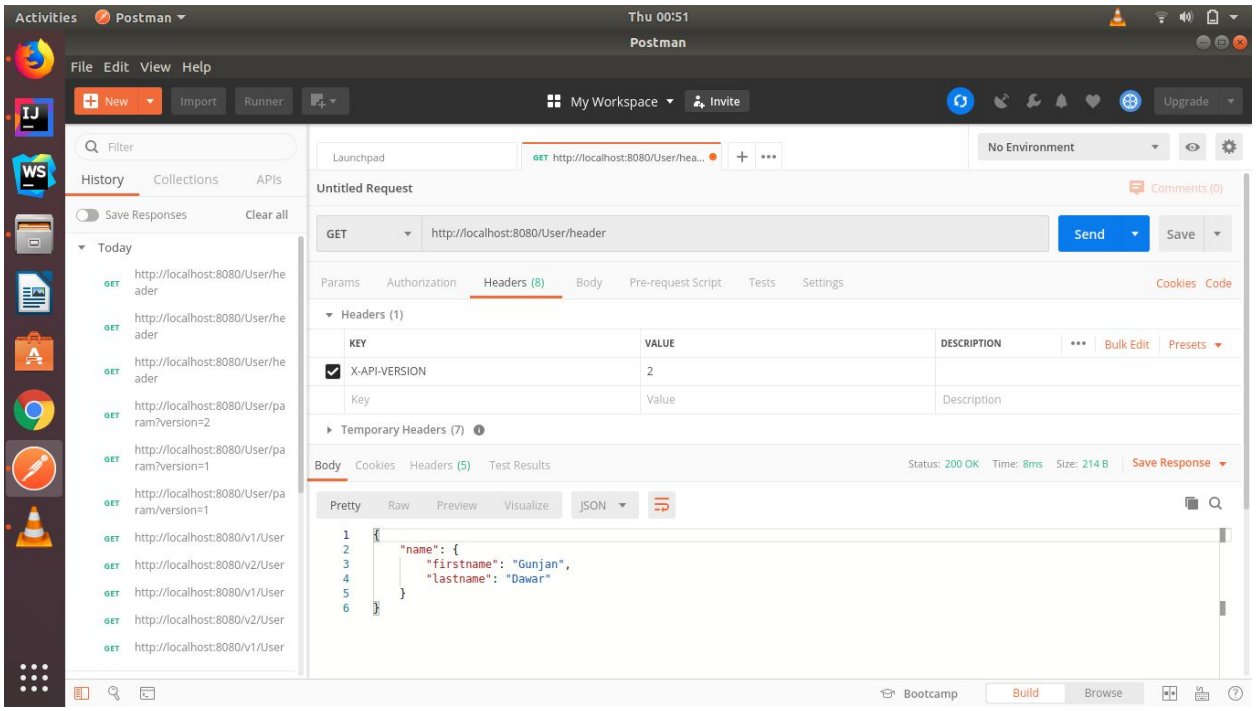

Activities
Postman
Thu 00:39
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/param?version=2
http://localhost:8080/User/param?version=1
http://localhost:8080/User/param/version=1
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
Yesterday
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/filtering
Launchpad
GET http://localhost:8080/User/para...
No Environment
GET
http://localhost:8080/User/param?version=2
Send
Save
Pretty
Raw
Preview
Visualize
JSON
{
"name": {
"firstname": "Gunjan",
"lastname": "Dawar"
}
}
Bootcamp
Build
Browse


Activities
Postman
Thu 00:39
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/param?version=1
http://localhost:8080/User/param/version=1
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
Yesterday
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/filtering
http://localhost:8080/hello-w
Launchpad
GET http://localhost:8080/User/para...
No Environment
GET
http://localhost:8080/User/param?version=1
Send
Save
Pretty
Raw
Preview
Visualize
JSON
{
"name": "Gunjan Dawar"
}
Bootcamp
Build
Browse

- Custom Header Versioning

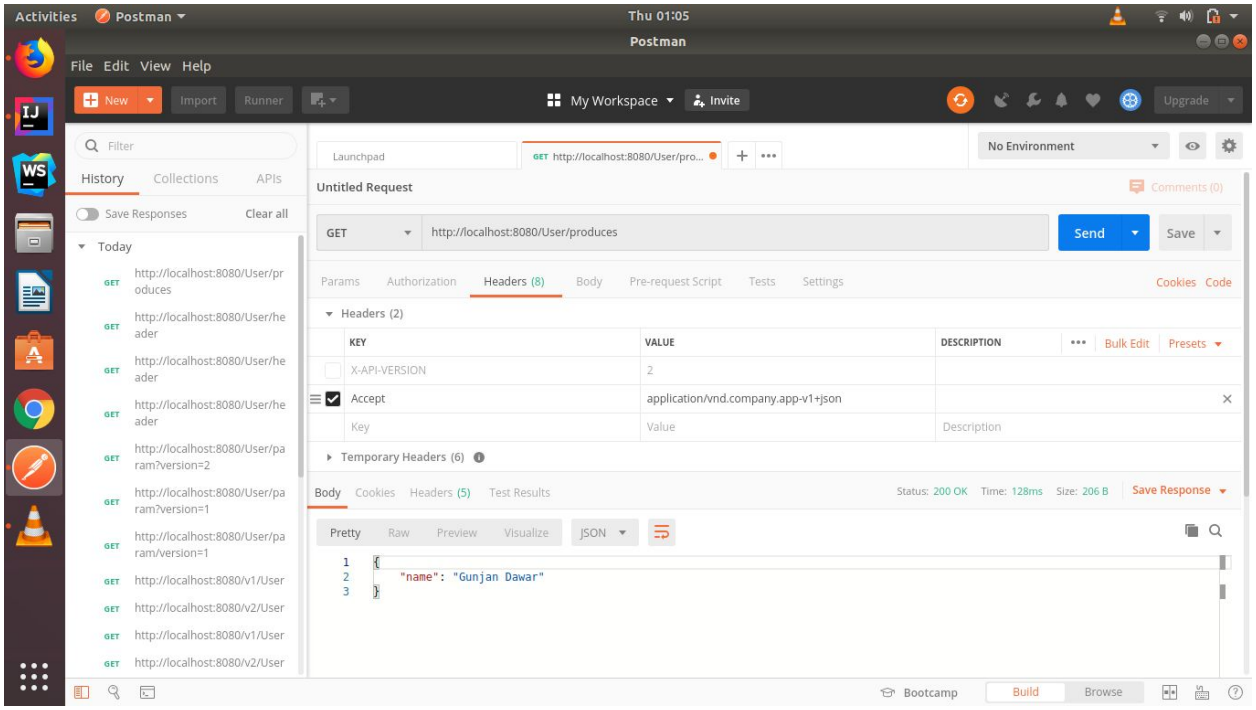
```
package com.example.RestfulWebService.Versioning;

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

@RestController
public class UserVersioningController {


  //Custom Header Versioning
    @GetMapping(value = "/User/header",headers = "X-API-VERSION=1")
   public UserV1 headerV1(){
      return new UserV1("Gunjan Dawar");

   }


   @GetMapping(value = "/User/header",headers = "X-API-VERSION=2")
   public UserV2 headerV2(){
      return new UserV2(new Name("Gunjan","Dawar"));
   }



}
```

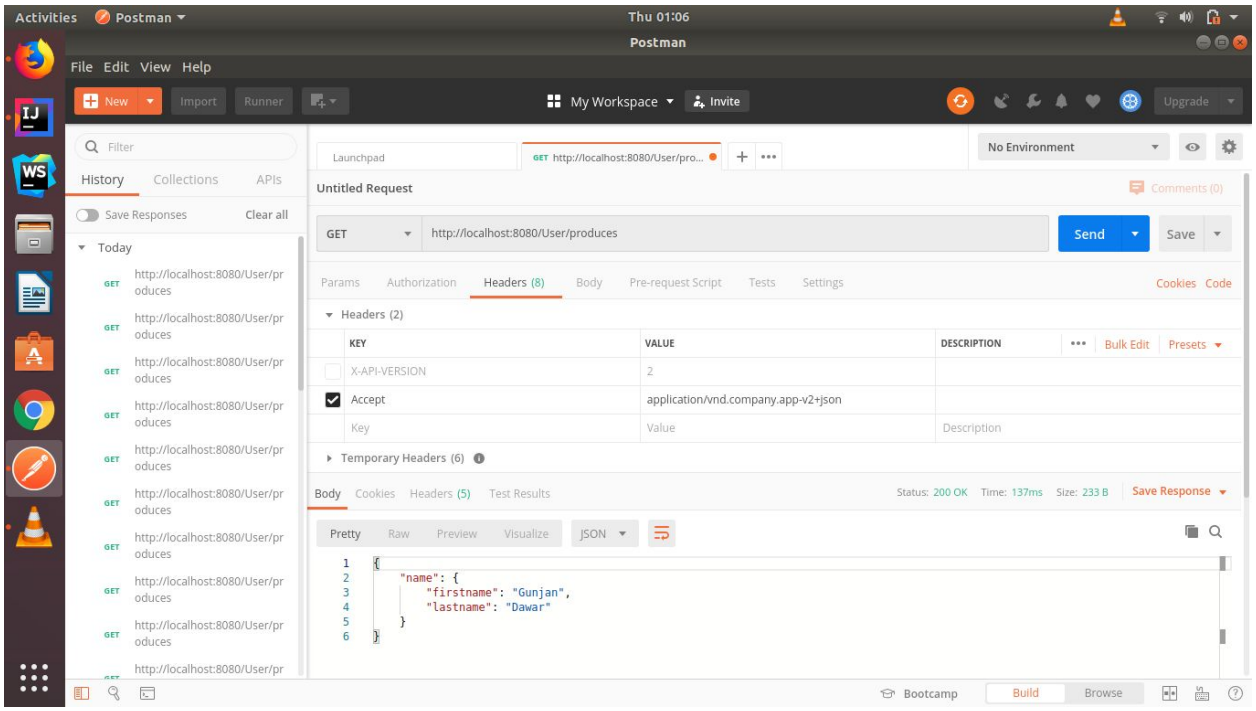

Activities
Postman
Thu 00:51
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/header
http://localhost:8080/User/header
http://localhost:8080/User/param?version=2
http://localhost:8080/User/param?version=1
http://localhost:8080/User/param/version=1
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
Yesterday
http://localhost:8080/filtering
Launchpad
GET http://localhost:8080/User/hea...
No Environment
Untitled Request
Comments (0)
GET
http://localhost:8080/User/header
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Headers (1)
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
X-API-VERSION
1
Key
Value
Description
Temporary Headers (7)
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 60ms
Size: 187 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": "Gunjan Dawar"
}
Bootcamp
Build
Browse


Activities
Postman
Thu 00:51
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/header
http://localhost:8080/User/header
http://localhost:8080/User/header
http://localhost:8080/User/param?version=2
http://localhost:8080/User/param?version=1
http://localhost:8080/User/param/version=1
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
Launchpad
GET http://localhost:8080/User/hea...
No Environment
Untitled Request
Comments (0)
GET
http://localhost:8080/User/header
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Headers (1)
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
X-API-VERSION
2
Key
Value
Description
Temporary Headers (7)
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 8ms
Size: 214 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": {
"firstname": "Gunjan",
"lastname": "Dawar"
}
}
Bootcamp
Build
Browse

- MimeType Versioning

```
package com.example.RestfulWebService.Versioning;

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

@RestController
public class UserVersioningController {


  //MimeType or Produces Versioning
  @GetMapping(value = "/User/produces",produces = "application/vnd.company.app-v1+json")
  public UserV1 producesV1(){
    return new UserV1("Gunjan Dawar");

  }


  @GetMapping(value = "/User/produces",produces = "application/vnd.company.app-v2+json")
  public UserV2 producesV2(){
    return new UserV2(new Name("Gunjan","Dawar"));
  }



}
```


Activities
Postman
Thu 01:05
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/produces
http://localhost:8080/User/header
http://localhost:8080/User/header
http://localhost:8080/User/header
http://localhost:8080/User/param?version=2
http://localhost:8080/User/param?version=1
http://localhost:8080/User/param/version=1
http://localhost:8080/v1/User
http://localhost:8080/v2/User
http://localhost:8080/v1/User
http://localhost:8080/v2/User
Launchpad
GET http://localhost:8080/User/pro...
No Environment
Untitled Request
Comments (0)
GET
http://localhost:8080/User/produces
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Headers (2)
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
X-API-VERSION
2
Accept
application/vnd.company.app-v1+json
Key
Value
Description
Temporary Headers (6)
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 128ms
Size: 206 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": "Gunjan Dawar"
}
Bootcamp
Build
Browse


Activities
Postman
Thu 01:06
Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
Save Responses
Clear all
Today
http://localhost:8080/User/produces
Launchpad
GET http://localhost:8080/User/pro...
No Environment
Untitled Request
Comments (0)
GET
http://localhost:8080/User/produces
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Headers (2)
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
X-API-VERSION
2
Accept
application/vnd.company.app-v2+json
Key
Value
Description
Temporary Headers (6)
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 137ms
Size: 233 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": {
"firstname": "Gunjan",
"lastname": "Dawar"
}
}
Bootcamp
Build
Browse

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

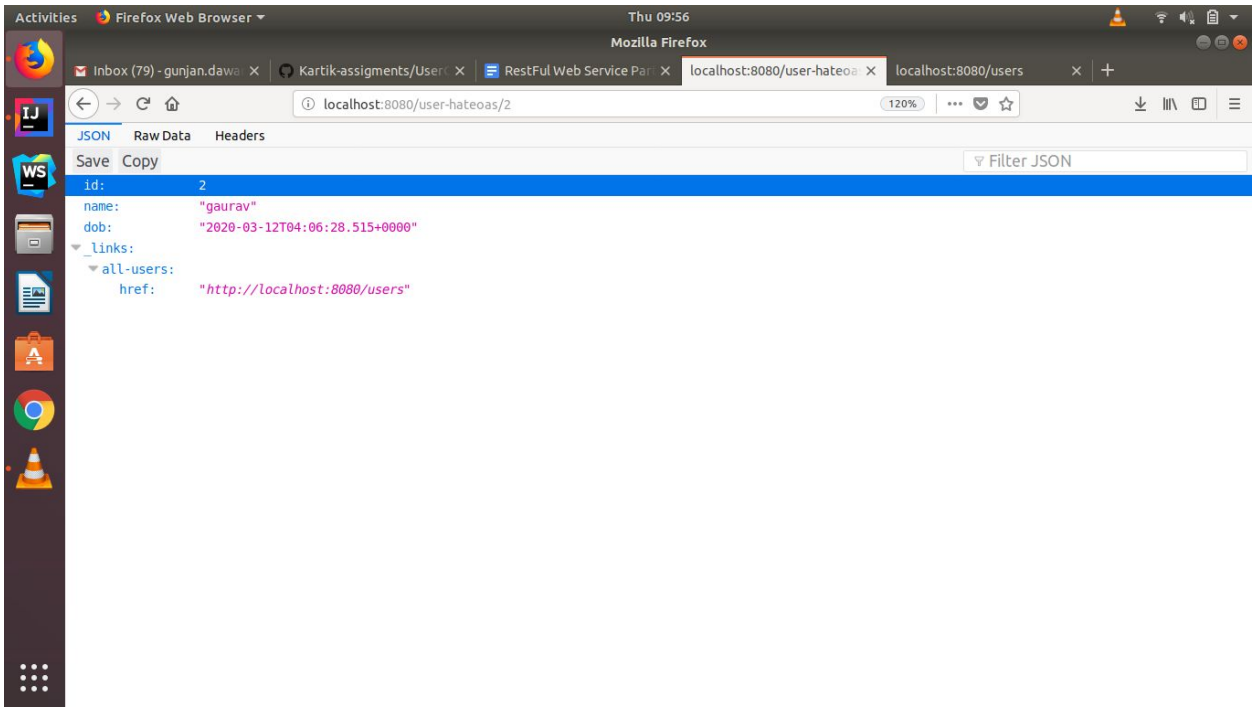
File=build.gradle

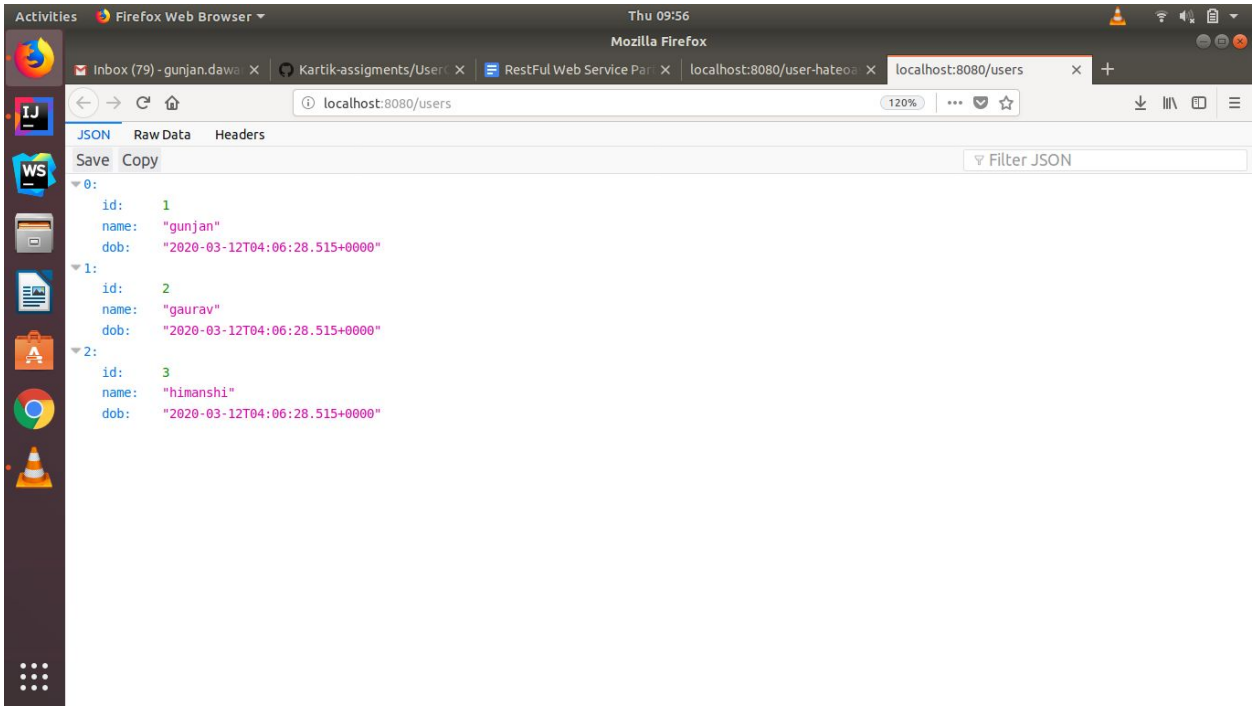
```
plugins {
  id 'org.springframework.boot' version '2.2.5.RELEASE'
  id 'io.spring.dependency-management' version '1.0.9.RELEASE'
  id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
  mavenCentral()
}

dependencies {

  implementation 'org.springframework.boot:spring-boot-starter-web'

compile group: 'org.springframework.boot', name: 'spring-boot-starter-hateoas'

  testImplementation('org.springframework.boot:spring-boot-starter-test') {
     exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
  }
}

test {
  useJUnitPlatform()
}
```

File=Userresource.java

```
package com.example.RestfulWebService.user;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
import org.springframework.hateoas.EntityModel;
import javax.validation.Valid;
import java.net.URI;
import java.util.List;

import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.methodOn;
```

```java
@RestController
public class userresource {

  @Autowired
  private Userdoa service;

  //Ques11 11. Configure hateoas with your springboot application. Create an api which returns User Details along with url to show all topics.
  @GetMapping(path = "/user-hateoas/{id}")
  public EntityModel<User> particularUser(@PathVariable int id)
  {
     User userDetail= service.findOne(id);
     if(userDetail==null)
        throw new UserNotFoundException("id/"+id);    // to generate an exception if user is not there
     EntityModel<User> model=new EntityModel<>(userDetail);
     WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).rettriveAllUSers());
     model.add(linkTo.withRel("all-users"));
     return model;
  }

}
```




Activities
Firefox Web Browser
Thu 09:56
Mozilla Firefox
Inbox (79) - gunjan.dawa
Kartik-assigments/User
RestFul Web Service Par
localhost:8080/user-hateo
localhost:8080/users
localhost:8080/users
120%
JSON
Raw Data
Headers
Save
Copy
Filter JSON
0:
id: 1
name: "gunjan"
dob: "2020-03-12T04:06:28.515+0000"
1:
id: 2
name: "gaurav"
dob: "2020-03-12T04:06:28.515+0000"
2:
id: 3
name: "himanshi"
dob: "2020-03-12T04:06:28.515+0000"