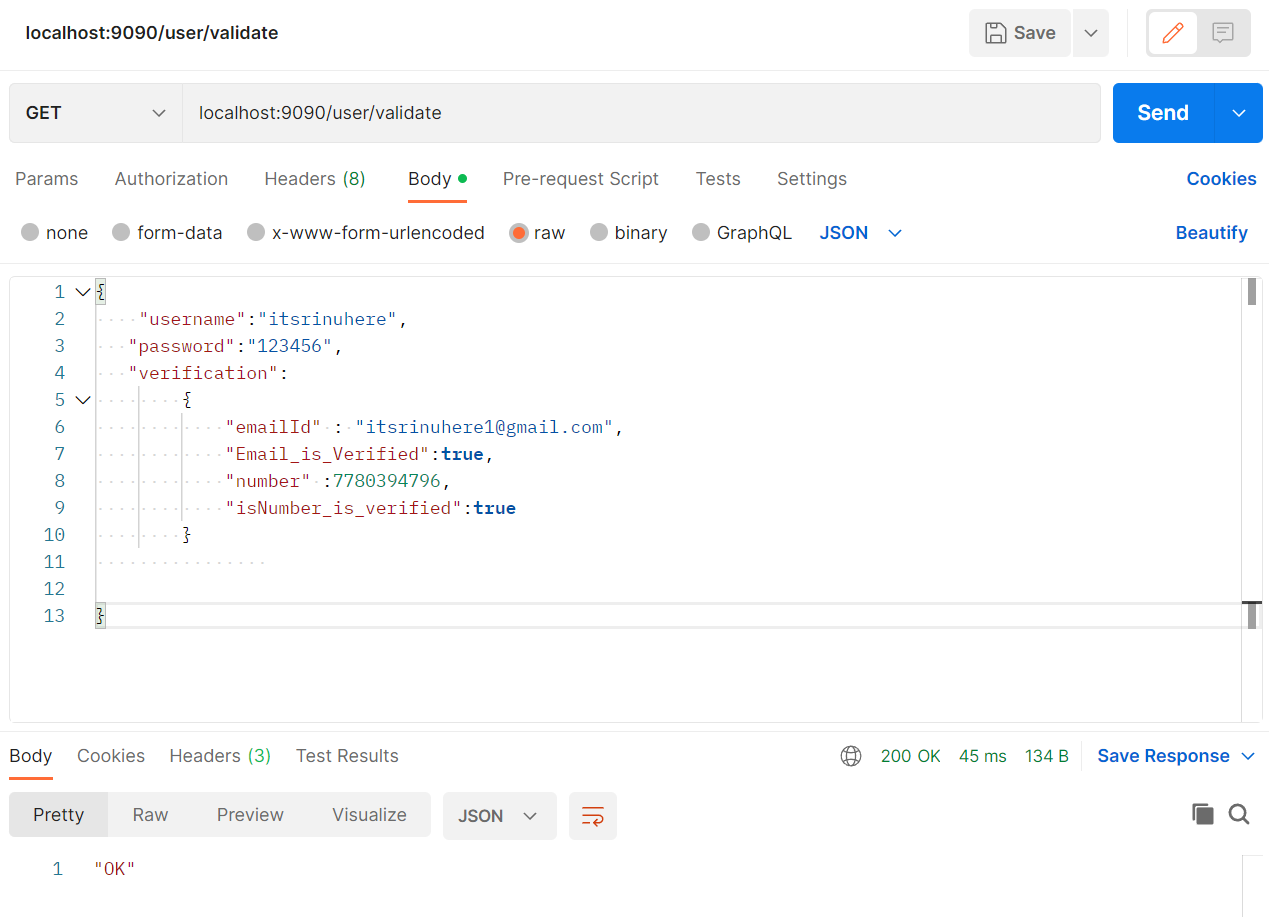
Project Weconnect

LiveReload server is running on port 35729



JSON:

{

    "username":"itsrinuhere",

   "password":"123456",

   "verification":

        {

            "emailId" : "itsrinuhere1@gmail.com",

            "Email\_is\_Verified":**true**,

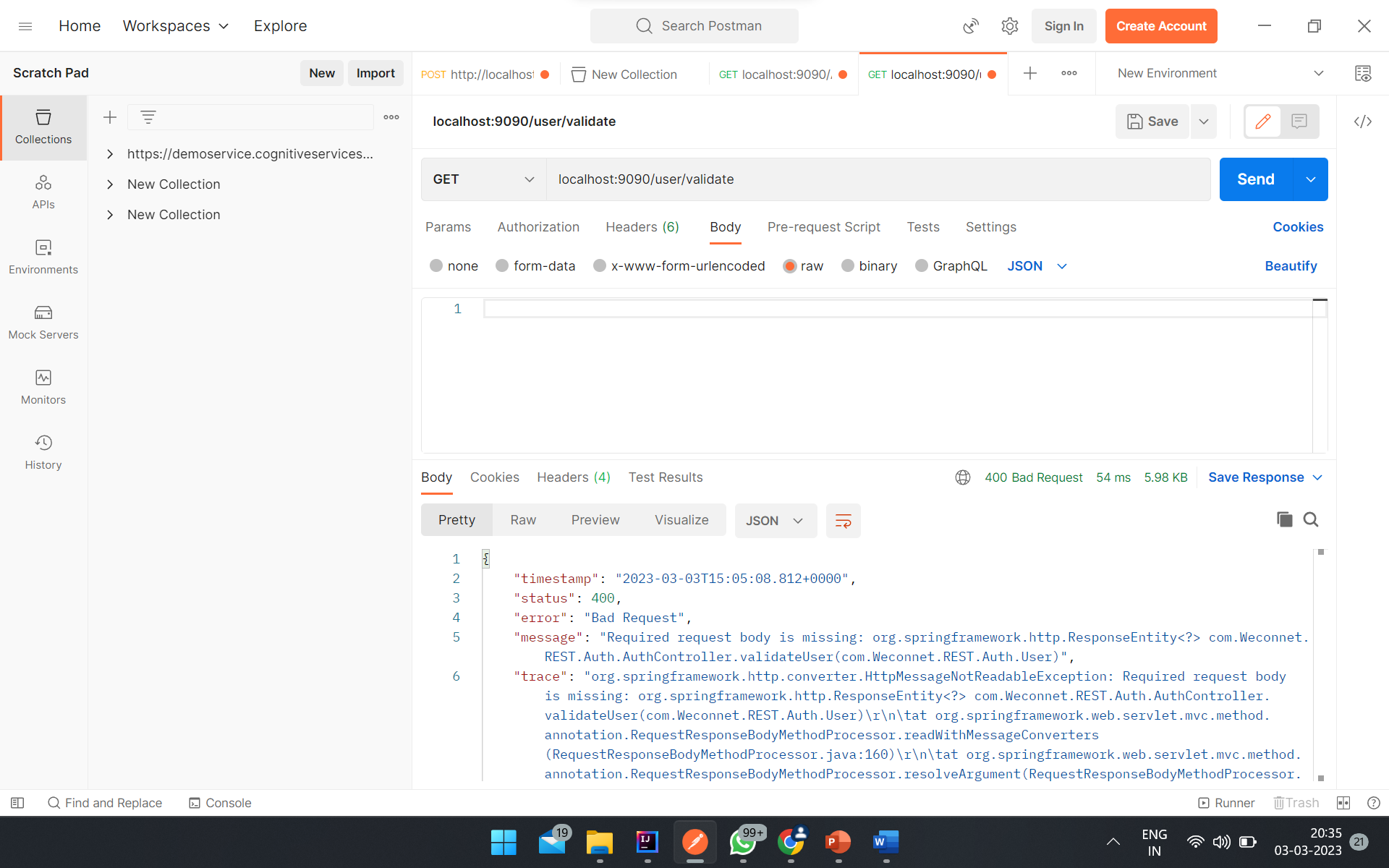
            "number" :7780394796,

            "isNumber\_is\_verified":**true**

        }

}

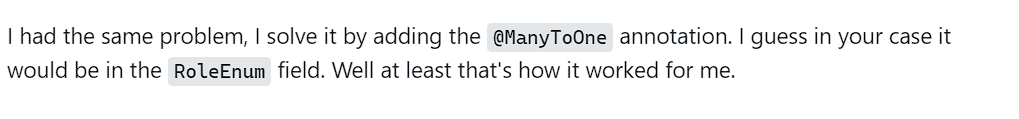
Sending null data:





DOCUMENTATION





public Set<Verification> getVerification() {  
 return verification;  
 }  
  
 public void setVerification(Set<Verification> verification) {  
 this.verification = verification;  
 }  
  
 @Column(name="verification")  
//@OneToMany(mappedBy="cart")  
//@NotNull  
//@OneToMany(mappedBy = "verification")  
@OneToMany(cascade=CascadeType.*ALL*, targetEntity=Verification.class)  
@JoinColumn(name="id")  
//@JoinColumn(name = "verification\_id", referencedColumnName = "id", nullable = false)  
Set<Verification> verification = new HashSet<>();

{

    "username":"itsrinuhere",

   "password":"123456",

   "verification":

        {

            "emailId" : "itsrinuhere1@gmail.com",

            "Email\_is\_Verified":**true**,

            "number" :7780394796,

            "isNumber\_is\_verified":**true**

        }

}

@Id

@GeneratedValue(strategy = GenerationType.SEQUENCE)

@Column(name = &quot;id&quot;, updatable = **false**, nullable = **false**)

**private** Long id;

84

+25

Yes, you can simply do it via wrapper class.

1) Create a Class to hold form data:

public class FormWrapper {

private MultipartFile image;

private String title;

private String description;

}

2) Create an HTML form for submitting data:

<form method="POST" enctype="multipart/form-data" id="fileUploadForm" action="link">

<input type="text" name="title"/><br/>

<input type="text" name="description"/><br/><br/>

<input type="file" name="image"/><br/><br/>

<input type="submit" value="Submit" id="btnSubmit"/>

</form>

3) Create a method to receive form's text data and multipart file:

@PostMapping("/api/upload/multi/model")

public ResponseEntity<?> multiUploadFileModel(@ModelAttribute FormWrapper model) {

try {

// Save as you want as per requiremens

saveUploadedFile(model.getImage());

formRepo.save(mode.getTitle(), model.getDescription());

} catch (IOException e) {

return new ResponseEntity<>(HttpStatus.BAD\_REQUEST);

}

return new ResponseEntity("Successfully uploaded!", HttpStatus.OK);

}

4) Method to save file:

private void saveUploadedFile(MultipartFile file) throws IOException {

if (!file.isEmpty()) {

byte[] bytes = file.getBytes();

Path path = Paths.get(UPLOADED\_FOLDER + file.getOriginalFilename());

Files.write(path, bytes);

}

}

POST /echo/post/json HTTP/1.1

Host: reqbin.com

Accept: application/json

Content-Type: application/json

Content-Length: 81

{

"Id": 78912,

"Customer": "Jason Sweet",

"Quantity": 1,

"Price": 18.00

}

Success full -> form a file upload

Check out

<https://techndeck.com/how-to-upload-file-using-apache-httpclient-in-java/>

<https://metamug.com/article/java/okhttp-post-request.html>

<https://stackoverflow.com/questions/29966123/httpurlconnection-proxy-strange-ioexception-error-writing-to-server>

<https://stackoverflow.com/questions/31131777/how-to-use-gson-to-convert-json-to-string>

🡪https://www.baeldung.com/httpurlconnection-post

* <https://www.baeldung.com/java-httpclient-post>
* <https://medium.com/javarevisited/building-simple-rest-api-client-in-plain-java-3ec300092c06>
* <https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-File-Upload-Servlet-Ajax-Example>
* <https://stackoverflow.com/questions/24890675/file-upload-using-rest-api-in-java>
* <https://stackoverflow.com/questions/49845355/spring-boot-controller-upload-multipart-and-json-to-dto>
* <https://springjava.com/spring-boot/json-object-with-file-upload-spring-rest-api>

