**Master SpringBoot**

**04-Validations and Global Exception Handling:**

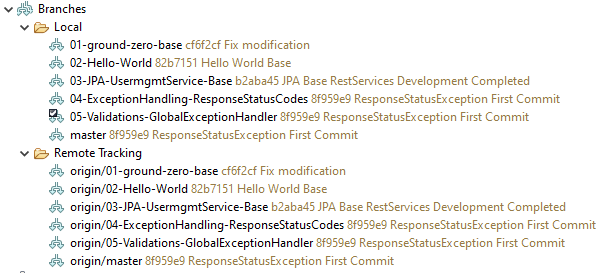
**Step-00: Create new git branch in local git repo and remote github repo**

* **Verify we are in master branch**
  + **git status**
* **Create new branch**
  + **git checkout –b 05-Validations-GlobalExceptionHandler**
* **Create new branch in remote github and setup upstream**
  + **git push –set-upstream origin 05-Validations-GlobalExceptionHandler**
* **Verify new branch in remote github from GIT Perspective**
  + [**https://github.com/jcoggins/springboot-buildingblocks**](https://github.com/jcoggins/springboot-buildingblocks)

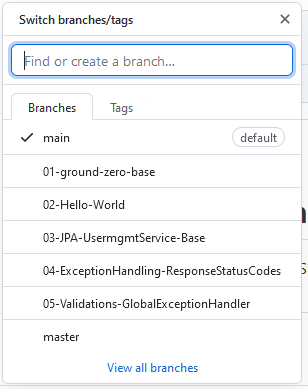
**git status git checkout … git push –set-upstream origin …**



**Check the Git Perspective:**



**Check remote Repository for branch 05:**



**Step-01: Implement Bean Validation**

* **Entity Layer**
  + **Implement validations on User Entity**
    - **@NotEmpty(message = “Username is Mandatory field. Please provide username”)**
    - **@Size(min=2, message=”FirstName should have at least 2 characters”)**
* **Test using POSTMAN**
  + **Service: Create User Service**
  + **Request body**
    - **For firstname, give only 1 character (“firstname”: “K”)**
    - **Send username as empty (“username”: “”)**
  + **Verify Response on POSTMAN (JPA exception will be thrown)**
* **Controller Layer**
  + **Enable Bean validation using @Valid**
* **Test using POSTMAN**
  + **Service: Create User Service**
  + **Request body**
    - **For firstname, give only 1 character (“firstname”: “K”)**
    - **Send username as empty (“username”: “”)**
  + **Verify Response**
    - **HTTP Status 400 Bad request – No Response body**
    - **MethodArgumentNotValidException resolved from logs**

**Add this dependency to POM.xml Otherwise, imports for validation are not available**

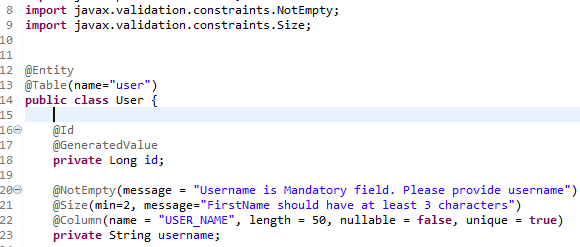
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

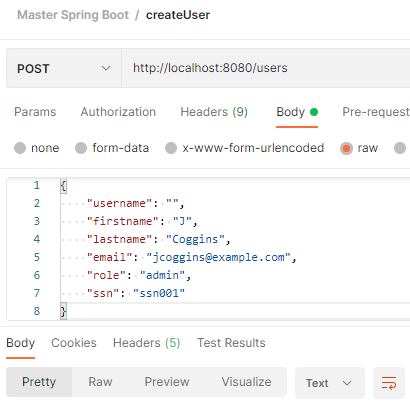
</dependency>

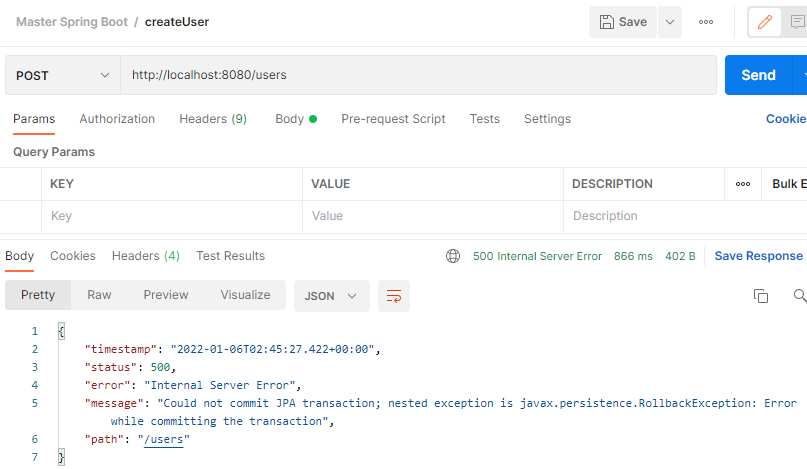
**Add the NotEmpty and Size constraints:**



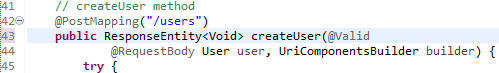
**Test with POSTMAN:**

**Go to createUser and change the username to empty, and the firstname to one character**

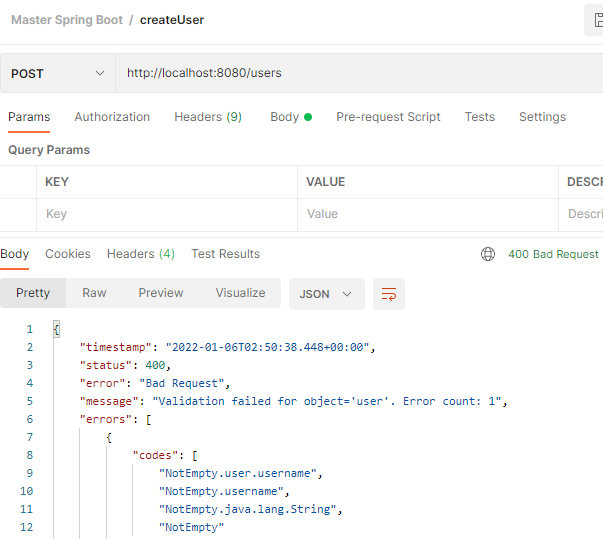




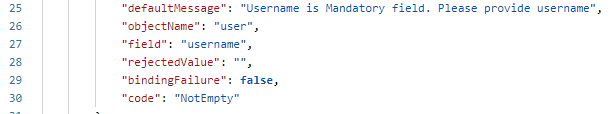
**Handle the JPA error with the Controller @Valid**

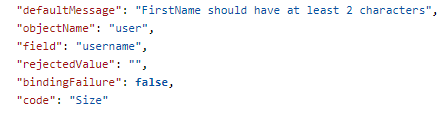


**Test again: Error is handled by Client as 400 error**



**Messages Provided:**





**StackTrace in console:**

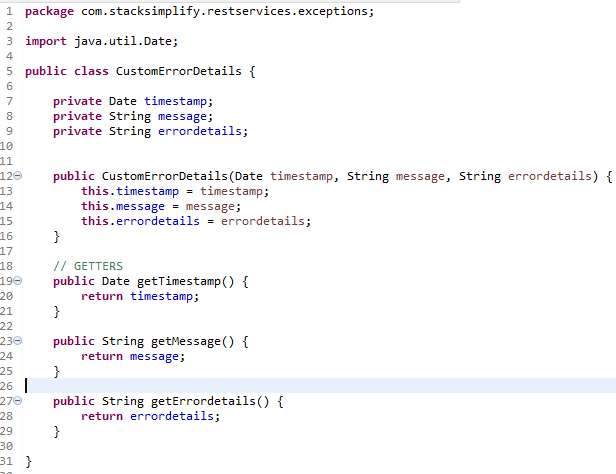
2022-01-05 21:50:38.443 WARN 19500 --- [nio-8080-exec-2] .w.s.m.s.DefaultHandlerExceptionResolver : Resolved [org.springframework.web.bind.MethodArgumentNotValidException: Validation failed for argument [0] in public org.springframework.http.ResponseEntity<java.lang.Void> com.stacksimplify.restservices.controllers.UserController.createUser(com.stacksimplify.restservices.entities.User,org.springframework.web.util.UriComponentsBuilder): [Field error in object 'user' on field 'username': rejected value []; codes [NotEmpty.user.username,NotEmpty.username,NotEmpty.java.lang.String,NotEmpty]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [user.username,username]; arguments []; default message [username]]; default message [Username is Mandatory field. Please provide username]] ]

**Now, reduce the default output provided by Spring, this the purpose of the Custom Global Exception Handler**

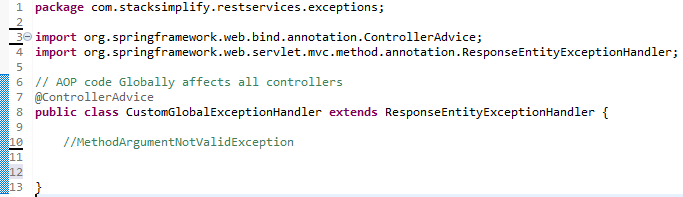
**Step-02: Implement Custom Global Exception Handler – MethodArgumentNotValidException**

* **Exception Layer – CustomErrorDetails**
  + **Create a new class CustomErrorDetails**
    - **Define Variables date, message, errordetails**
  + **Add Fields Constructor**
  + **Add Getters**
* **Exception Layer – CustomGlobalExceptionHandler**
  + **Create a new class CustomGlobalExceptionHandler**
  + **Extends ResponseEntityException**
  + **@ControllerAdvice**
    - **Global code that can be applied to a wide range of controllers**
  + **Implement & Override handleMethodArgumentNotValid from ResponseEntityExceptionHandler**
* **Test using POSTMAN**
  + **Service: Create User Service**
  + **Request body**
    - **For firstname, give only 1 character (“firstname”: “K”)**
    - **Send username as empty (“username”: “”)**
  + **Verify Response**
    - **Custom Error Response**
    - **HTTP 400 Bad Request**

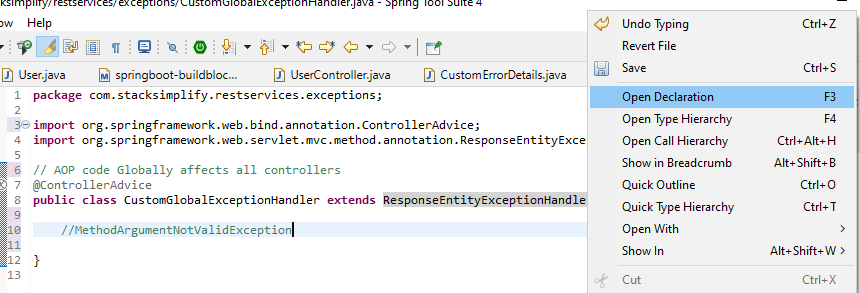
**Create CustomErrorInforDetails class**



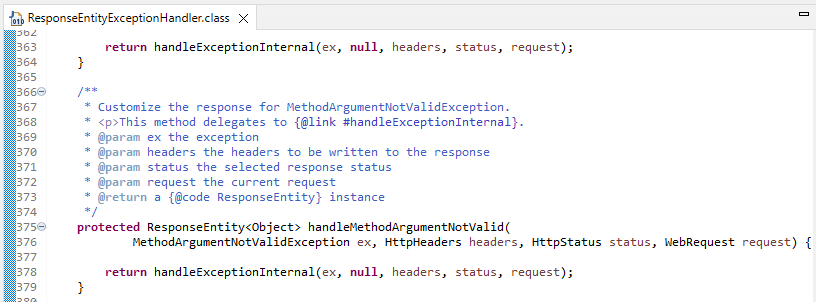
**Create the CustomGlobalExceptionHandler:**



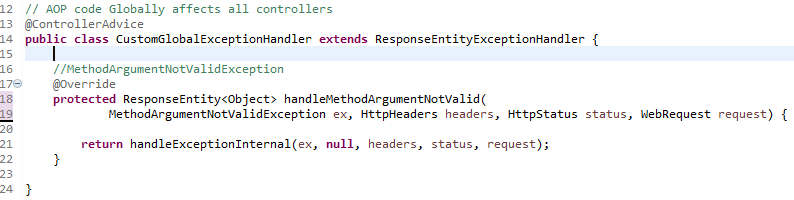
**Find the Method to Override:**

****

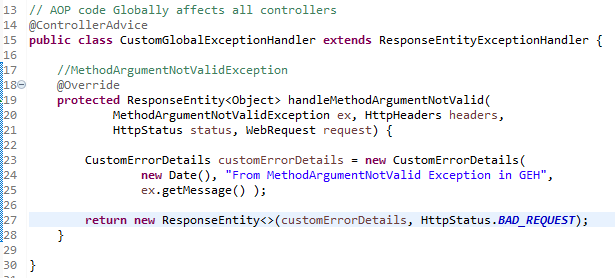
**Found MethodArgumentNotValidException - handleMethodArgumentNotValid**

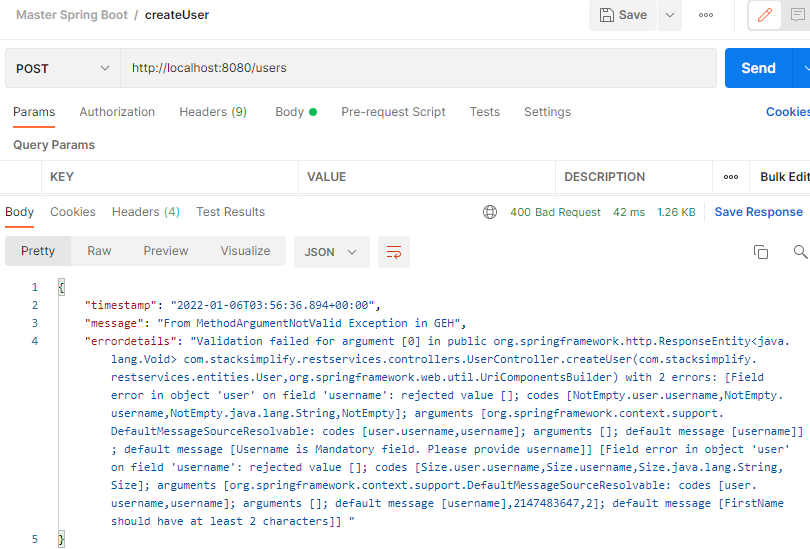


**Copy and paste into CustomErrorExceptionHandler**



**Fix the method:**



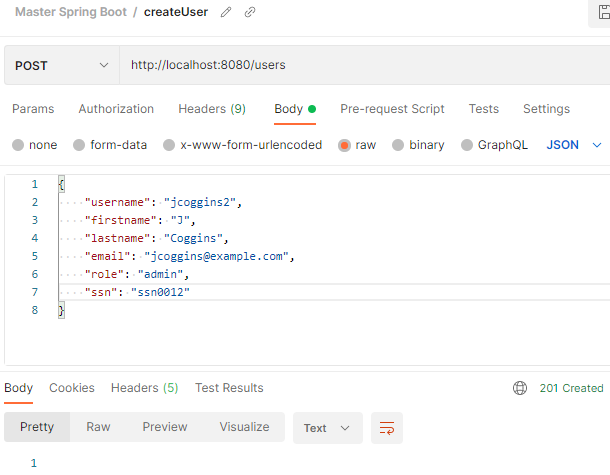
**Test with POSTMAN:**

**Error Output has been customized!!**

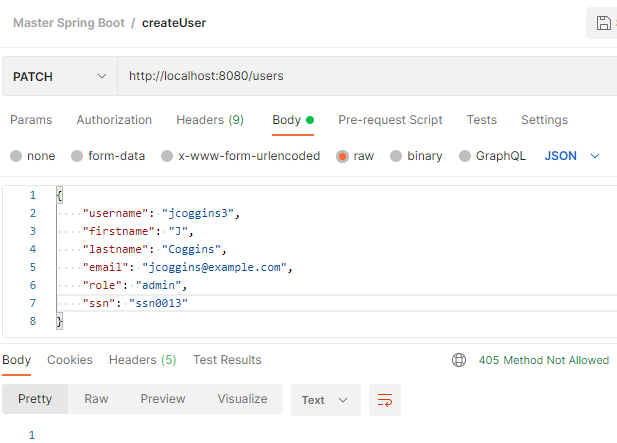
**Step-03: Implement “HttpRequestMethodNotSupportedException” in Custom Global Exception Handler**

* **Test using Postman**
  + **Provide PATCH method for create user**
  + **Verify response code and body**
* **Exception Layer - CustomGlobalExceptionHandler**
  + **Implement & Override handleMethodArgumentNotValid for ResponseEntityExceptionHandler**
* **Test using Postman** 
  + **Provide PATCH method for createUser**
  + **Verify response code for body**

**Create a new user successfully: The status code is 201**



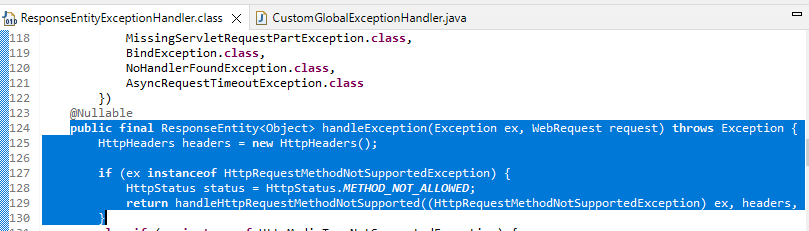
**Now, switch the method to PATCH method to create a different user**



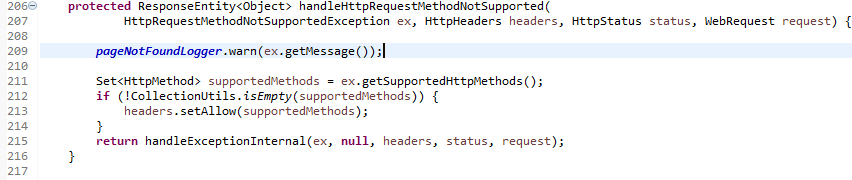
**The status code 405 is expected, method has not be implemented!**

**We need to customize the output with a message error!**

**Find the Exception:**



**Find the handler: Copy the method and customize it**



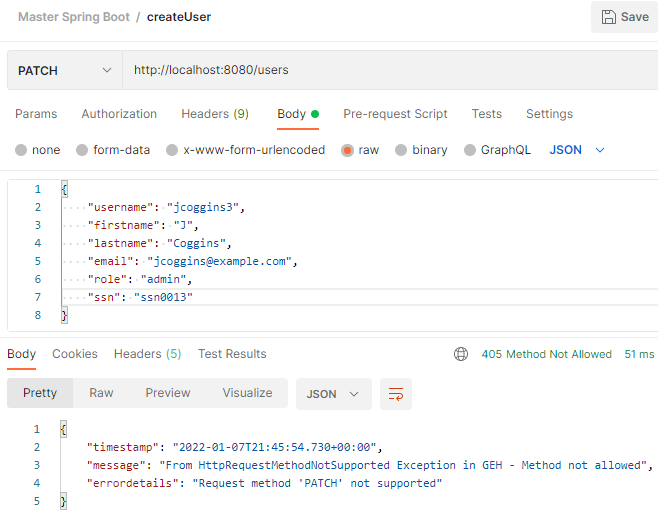
**Paste the method into the CustomGlobalExceptionHandler**



**Edit the method with @Override**



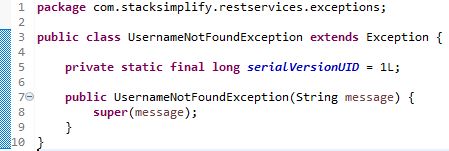
**Test with POSTMAN again:**



**Step-04: Implement ExceptionHandler for custom exception link “UserNameNotFoundException”**

* **Exception Layer**
  + **extends Exception**
* **Controller Layer**
  + **Throw UserNameNotFoundException if that user does not exist in Repository**
* **Test using Postman** 
  + **Method: GET**
  + **URI: http://localhost:8080/users/byusername/abcd**
  + **Verify default spring Exception response code HTTP 500**
* **Exception Layer – CustomGlobalExceptionHandler**
  + **Create handleUserNameNotFoundException class**
  + **Annotation it with @ExceptionHandler**
* **Test using POSTMAN**
  + **Method: GET**
  + **URI: http://localhost:8080/users/byusername/abcd**
  + **Verify the Response Body and HTTP Status Code**

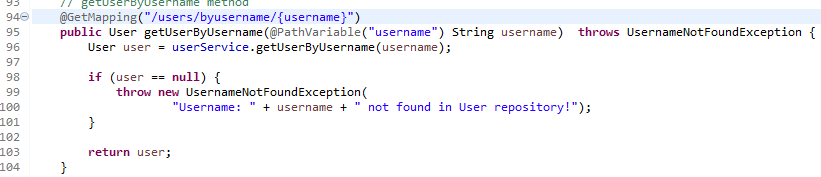
**Create the Custom Exception:**



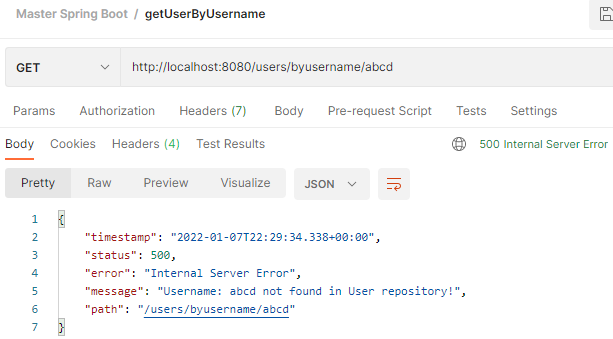
**Edit the getUserByUserName method in Service Layer**



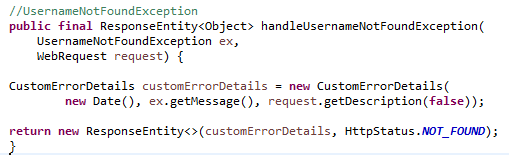
**Edit the getUserByUserName method in Controller Layer with message!!**



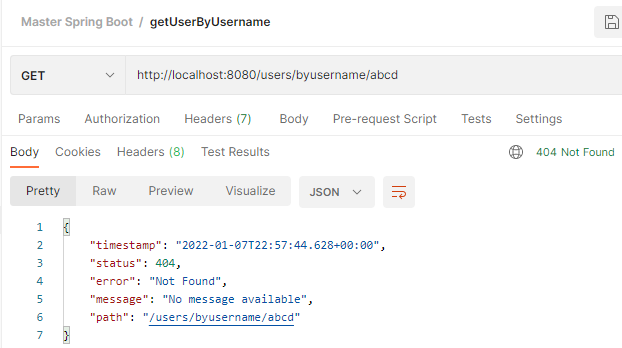
**Test with POSTMAN:**



**Now, use a custom handler:**



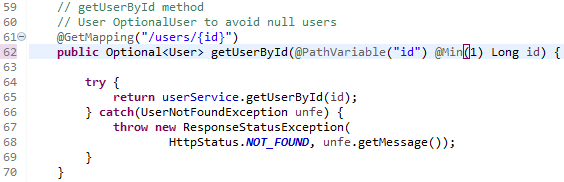
**Test with POSTMAN:**



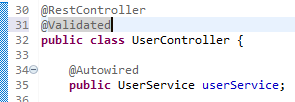
**Step-05: Path Variables Validation & Handling ConstraintViolationException using CustomGlobalExceptionHandling**

* **Controller Layer**
  + **getUserById method: Add @Min(1) for Path Variable**
  + **Add @Validated annotation to UserController class**
* **Test using Postman** 
  + **Method: GET**
  + **URI: http://localhost:8080/users/0**
  + **Verify default spring Exception (response code HTTP 500)**
* **Exception Layer - CustomGlobalExceptionHandler**
  + **Create handleConstraintViolationException class**
  + **Annotate it with @ExceptionHandler**
* **Test using Postman** 
  + **Method: GET**
  + **URI: http://localhost:8080/users/0**
  + **Verify the Response Body and HTTP Status Code**

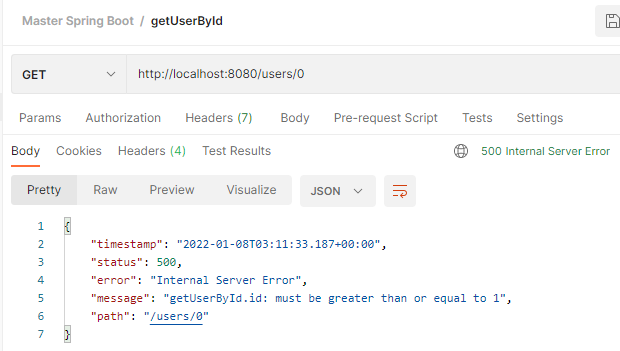
**Using the UserController class add the @Min(1) to getUserById:**



**Add @Validated**

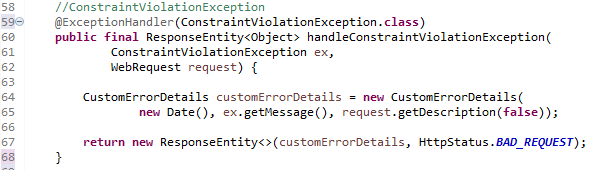


**Test with POSTMAN:**

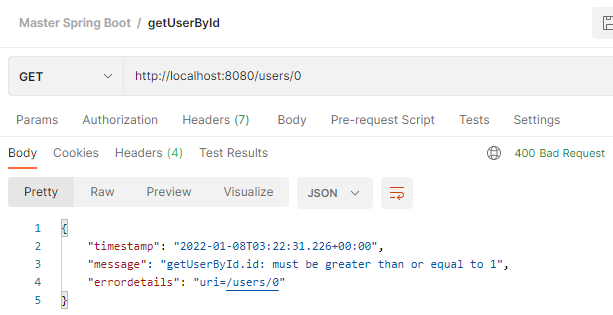


**500 status code is in correct, it should be a Client Error 4XX**

**Add the handler:**



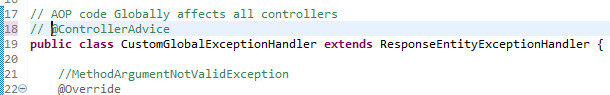
**Test with POSTMAN:**



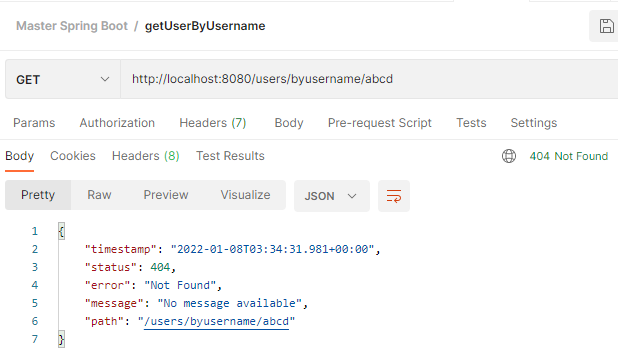
**Step-06: Implement Global Exception Handling using RestControllerAdvice**

* **Exception Layer - CustomGlobalExceptionHandler**
  + **Comment @ControllerAdvice**
  + **Test to ensure controller advice is not in action for getUserByUsername**
* **Exception Layer** 
  + **Create new class GlobalRestControllerAdviceExceptionHandler**
  + **Annotate with @RestControllerAdvice**
  + **Handle UsernameNotFoundException**
    - **Create a method usernameNotFound**
    - **Annotate it with @ExceptionHandler(UsernameNotFoundException.class)**
    - **Annotate it with @ResponseStatus(HttpStatus.NOT\_FOUND)**
* **Test using Postman** 
  + **Method: GET**
  + **URI: http://localhost:8080/users/byusername/abcd**
  + **Verify the Response Body and HTTP Status Code**

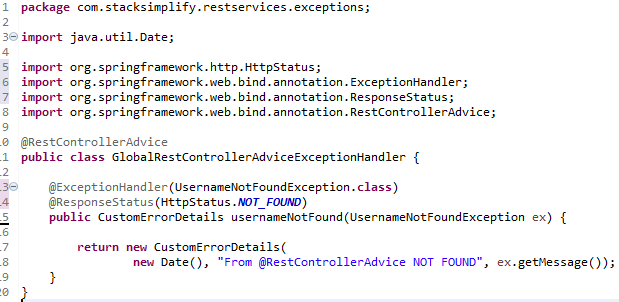
**Comment out the @ControllerAdvice**



**Test with Postman:**



**Implement the GlobalRestControllerAdviceExceptionHandler:**



**Test with Postman:**

