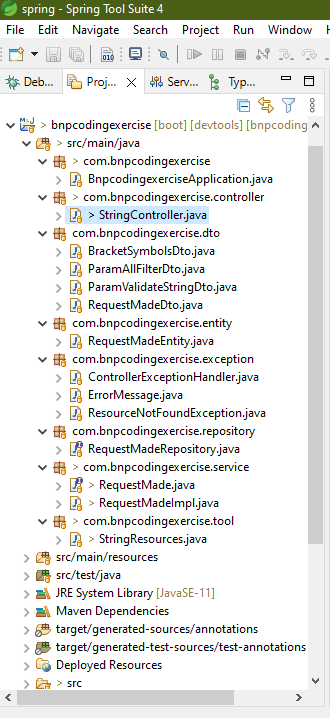
**Correctness of bracket expressions Documentation**

**Note:** The exercise was developed using STS 4, maven project in Spring boot, for run the project clone de repo <https://github.com/noeldmb/bnpcodingexercise> open the project with IDE (STS 4, Eclipse, IntelliJ IDEA, VS code or any ide that be able for it), runs maven in order to update dependencies, and run de project. The endpoints were tested with Postman.

The endpoints created are related further in this document, indicating the url for each endpoint, and showing some pictures

The application has the following structure:

1-***package com.bnpcodingexercise***: it is the root packet of the app, in this packet is locate the entry class of the app.

2-***package com.bnpcodingexercise.controller***: this package is for holding all the controllers declared in the app, in this case, there is 1 controller only.

3-***package com.bnpcodingexercise.dto***: this package is used for holding all “Dto” object that transport data, like parameters, and uses for response data to the client, in this case, Postman app.

4-***package com.bnpcodingexercise.entity***: this package is for Entity classes, used for Spring boot JPA handling data into Data Bases in this case H2 Data Base.

5-***package com.bnpcodingexercise.exception***: this package contains all the classes needed for handling exceptions in the app.

6-***package com.bnpcodingexercise.repository***: package for holding the repository Interface, this repository is used for Spring boot JPA to interact with data in the Data Base

7-**package com.bnpcodingexercise.service**: this package contains all interfaces and classes that normally implement the corresponding interface, used for implementing the business logic needed according to the case.

8-***package com.bnpcodingexercise.tool***: package for holding classes of general use.

The exercise was developed according to following requirement:

***“Create a Spring-based web REST service to check the string for the correctness of its bracket expressions. The service should support custom symbols to treat as "open bracket" and "close bracket". For example, "(" - ")", "{" - "}".***

***The service should:***

*1. Accept a string, pair of symbols which are to be considered as open-closed brackets and username*

*2. Validate correctness of the requested strings and return the validation result back to the requested.*

*1. All symbols which match the open bracket should have the corresponding close ones. No unclosed open bracket symbols.*

*2. All close bracket symbols should have a corresponding open ones located prior to it in the requested string.*

*3. Each request should be stored to a persistence storage: string, username, used symbols and result.*

*4. A separate endpoint to get the data from the storage should be supported with the following filtering (filters may be combined together). API on your choice:*

*1. Get all strings by user*

*2. Get all strings by result (valid/not-valid)*

*3. Get all strings by pairs of "open bracket" and "close bracket" symbols.*

*Example:*

*POST /bracket-service/validate*

*{*

*"string": "(testing (usual brackets) done)",*

*"username": "Anton",*

*"bracketSymbols": {*

*// symbol to use as "open-bracket"*

*"open": "(",*

*// symbol to use as "close-bracket"*

*"close": ")"*

*}*

*}*

*POST /bracket-service/validate*

*{*

*"string": "){}(",*

*"username": "Anton",*

*"bracketSymbols": {*

*// symbol to use as "open-bracket"*

*"open": "{",*

*// symbol to use as "cloase-bracket"*

*"close": "}"*

*}*

*}*

**Endpoints built**

1. <http://localhost:8080/bracket-service/validate> used for validate and string, it is a http POST request, the parameters are passed in red border, and the result in green borther, parameters are:

{

"string": "){}(",

"username": "user1",

"bracketSymbols": {

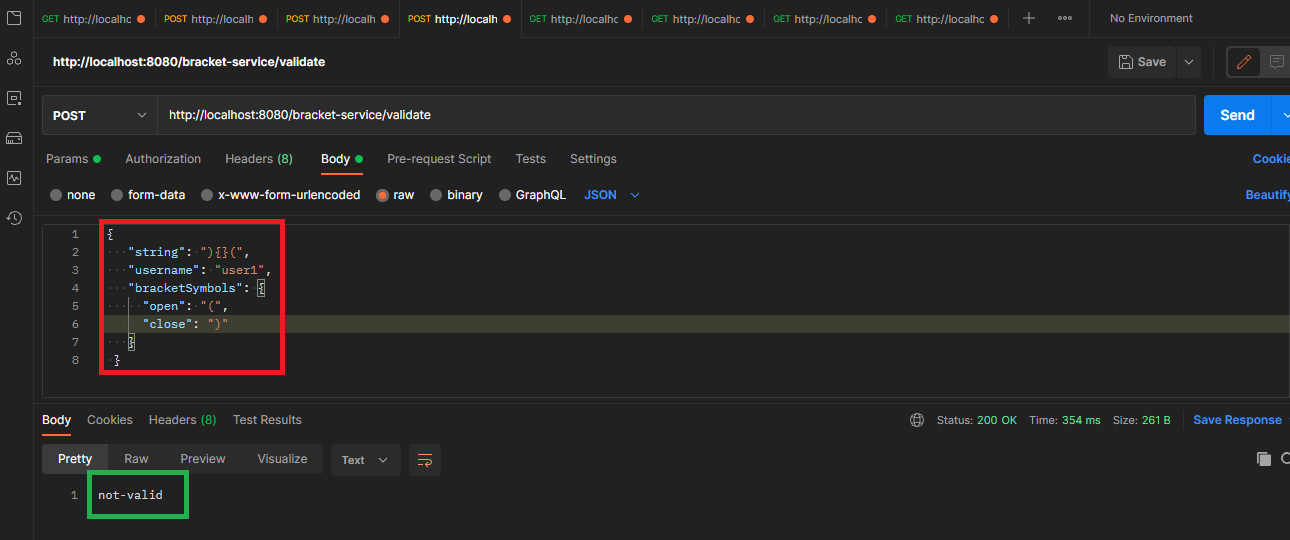
"open": "(",

"close": ")"

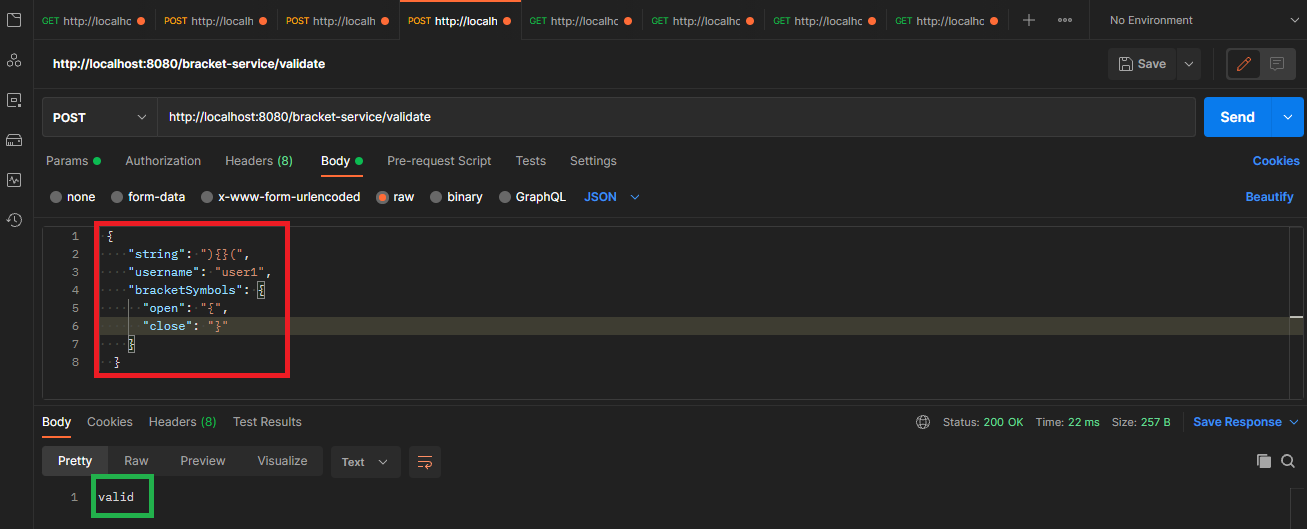
}

}

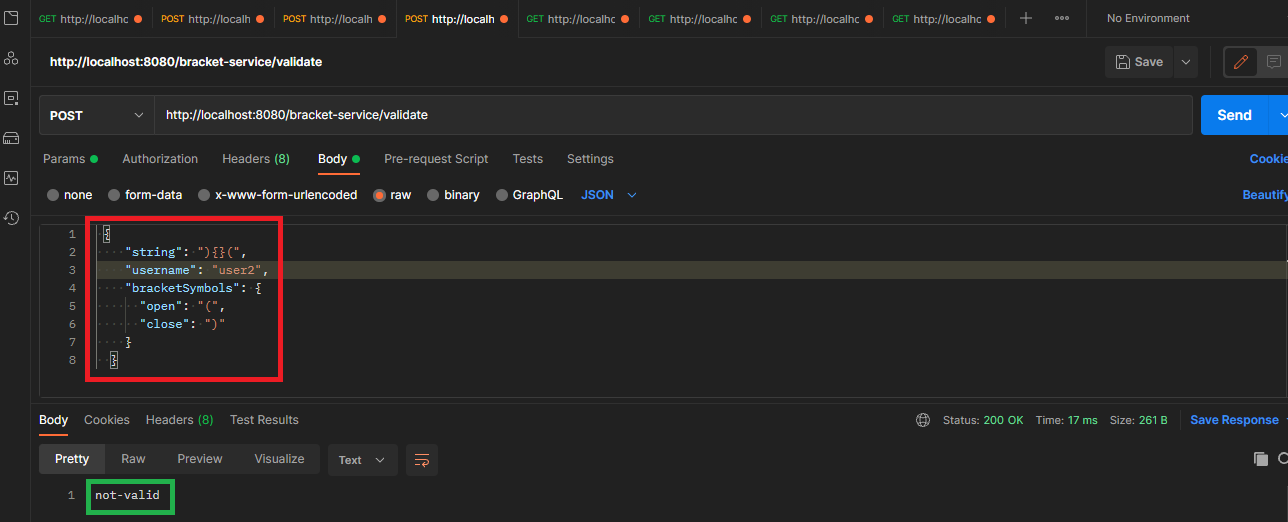
Answer: not-valid “()” are not correct in "string": "){}("



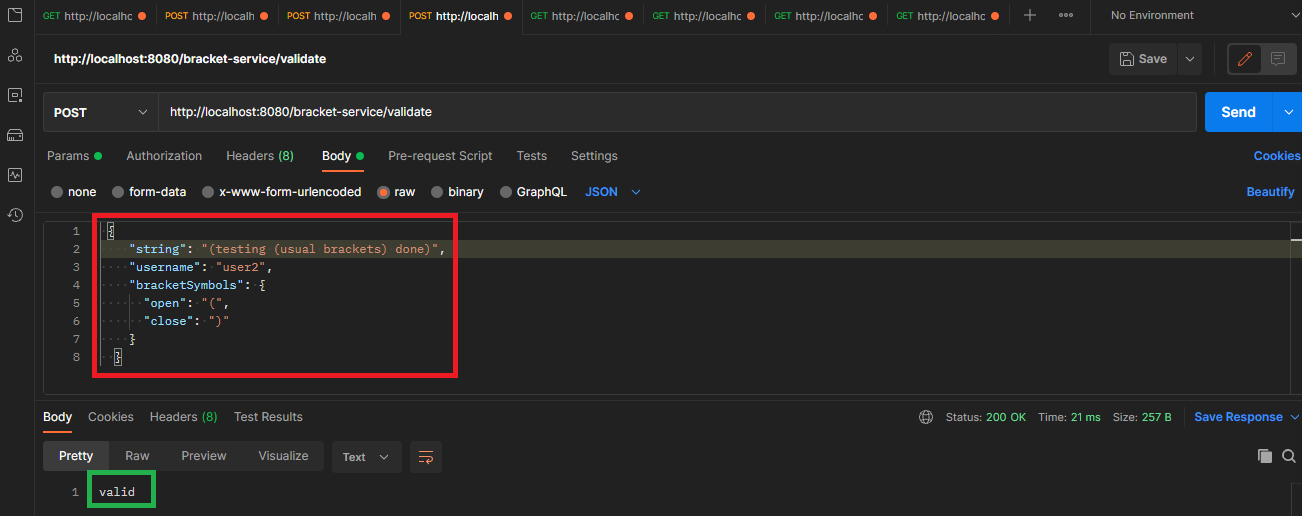
Answer: valid “{}” are correct in "string": "){}("



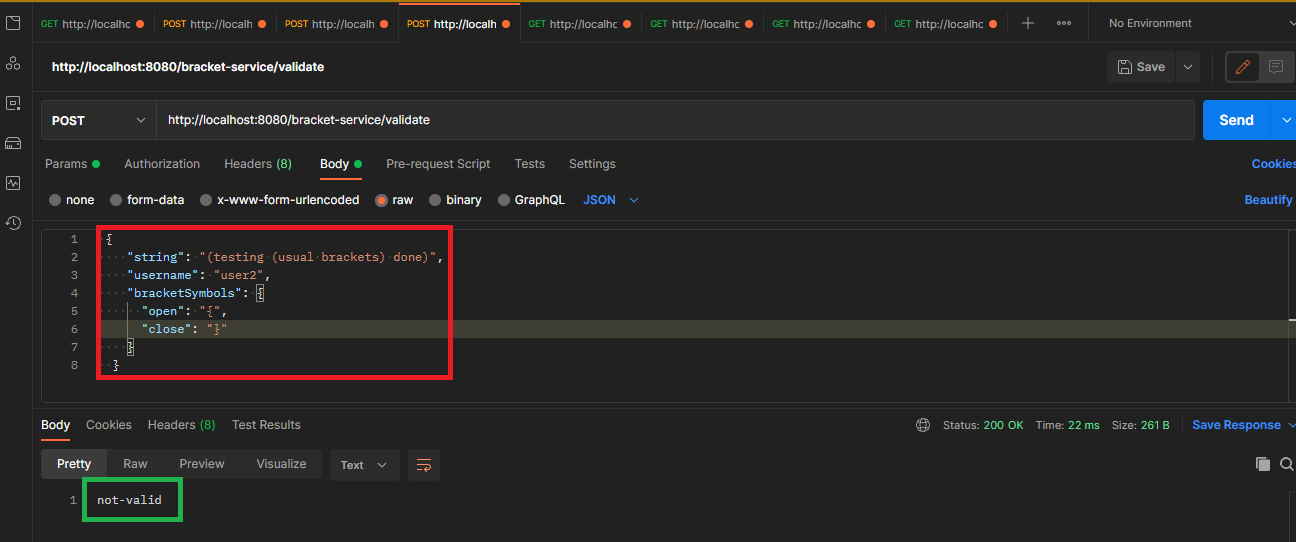
Answer: different user “user2”, the “()” are not in correct order



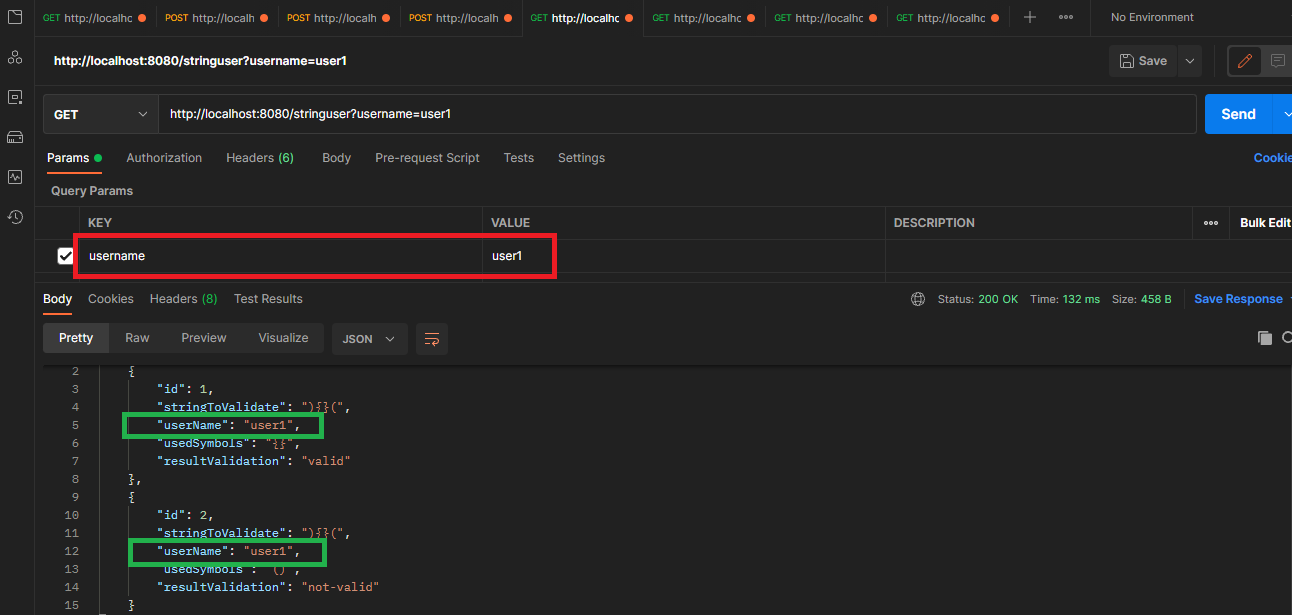
Answer: the “()” are in correct order in the string.



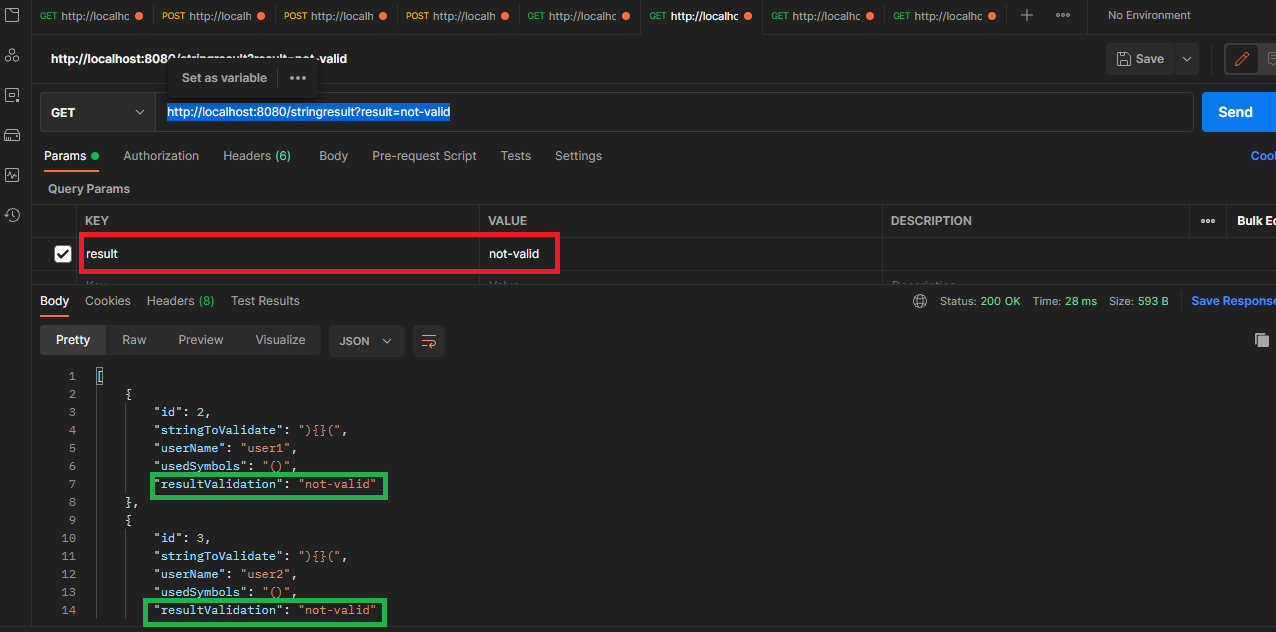
Answer: the “{}” not exists in the string



1. <http://localhost:8080/stringuser?username=user1> endpoint used for search by “username”, in this case show a List with al request made by user “user1”



1. <http://localhost:8080/stringresult?result=not-valid> endpoint used for retrieve validations by their result “valid” or “not-valid”



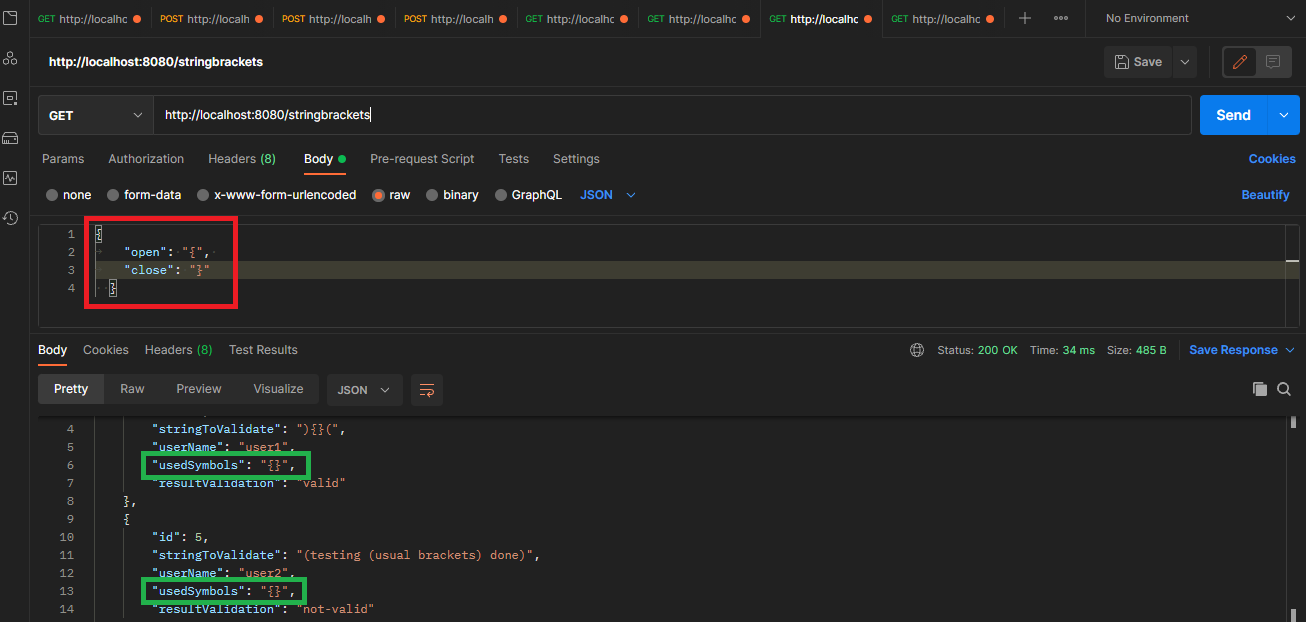
1. <http://localhost:8080/stringbrackets> endpoint used for retrieve validations by type of brackets, in this case shown e list of all validations made using “{}” brackes. Param in the body of the request.

{

"open": "{",

"close": "}"

}



1. <http://localhost:8080/stringallfilters> endpoint used for retrieve by combination filter’s criterial, if all criterial are blank, show all the validations made, the params are passed in an object

{

"result": "",

"username": "",

"bracketSymbols": {

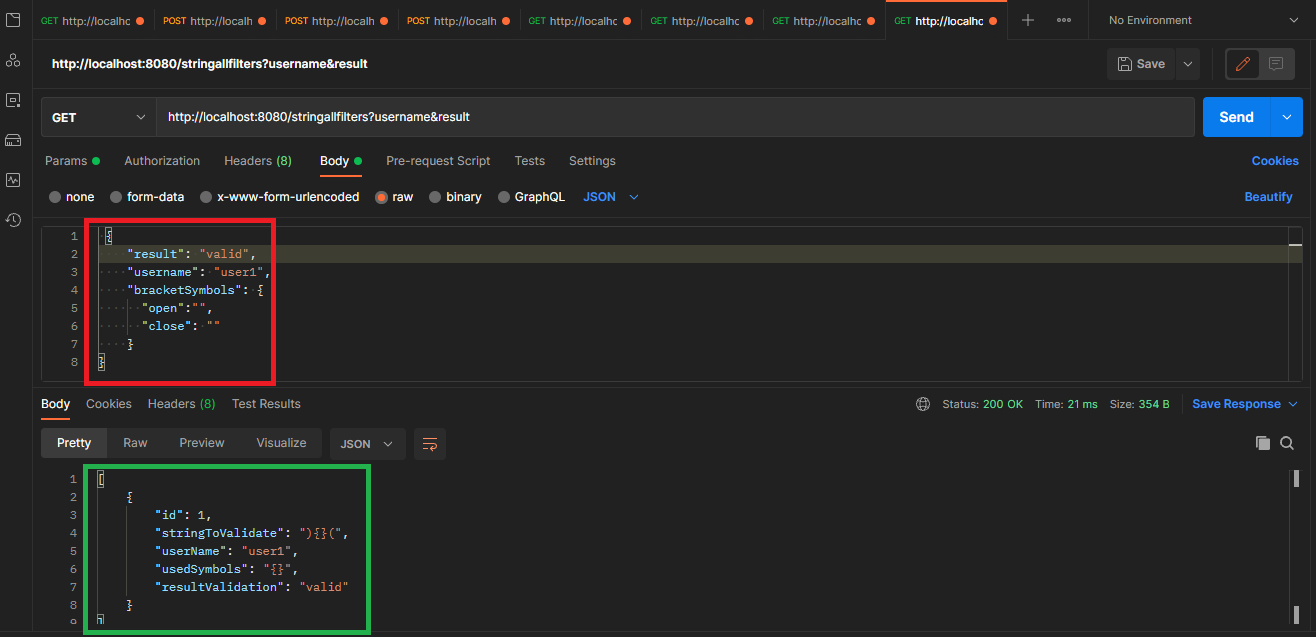
"open":"",

"close": ""

}

}

Answer: shown all the validations made by “user1” and the “result” was “valid”



Answer: shown all validation done.

