**REST API part 2**

[**http://localhost:8080/v2/api-docs**](http://localhost:8080/v2/api-docs)

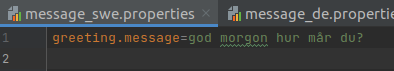
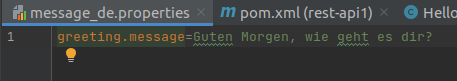
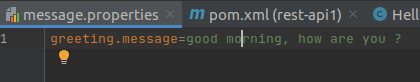
**1. Add support for Internationalization in your application allowing messages to be shown in English, German and Swedish, keeping English as default.**

helloworldcontroller.java

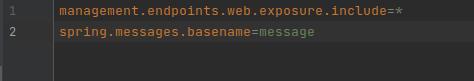
****

main class (for LocaleResolver)****

message.properties

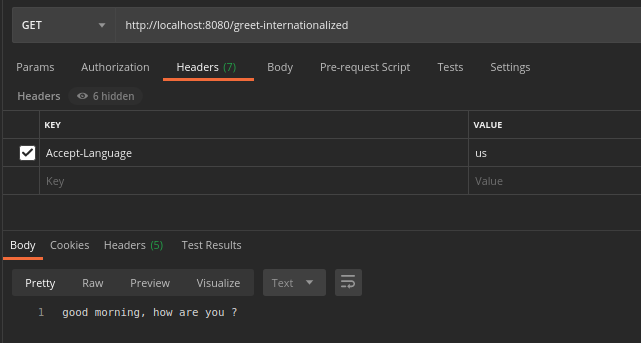
****

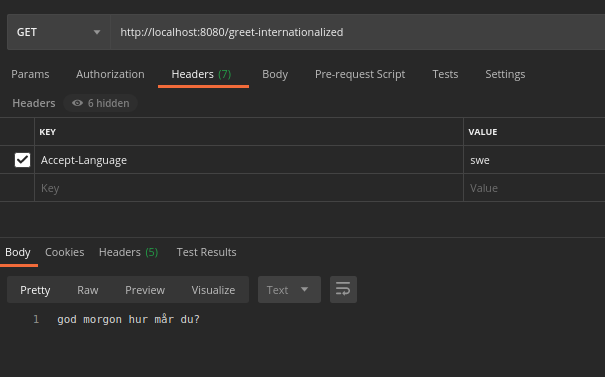
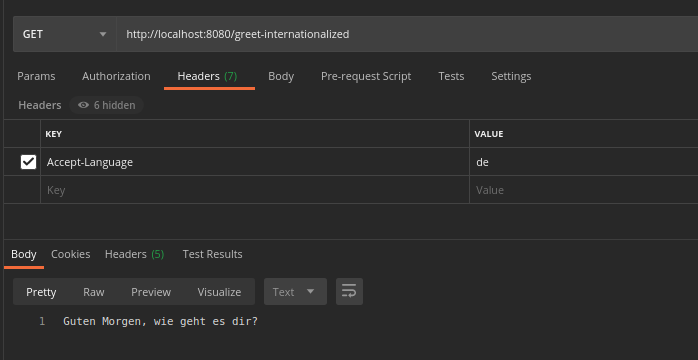
Application.properties

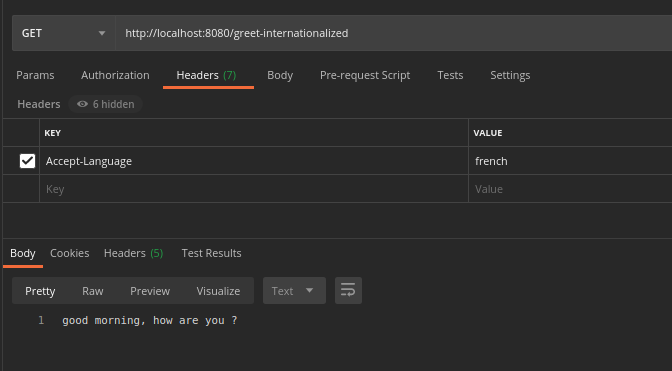
****

**Output:**

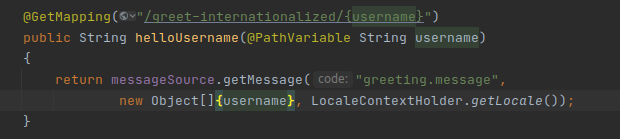
[**http://localhost:8080/greet-internationalized**](http://localhost:8080/greet-internationalized) **(deafult)**

****

****

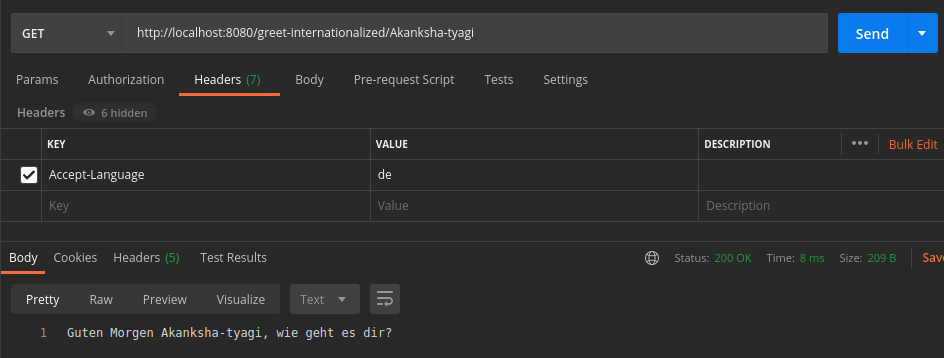
****

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

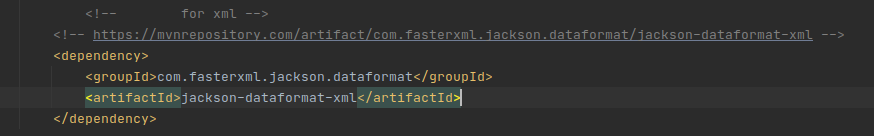
****

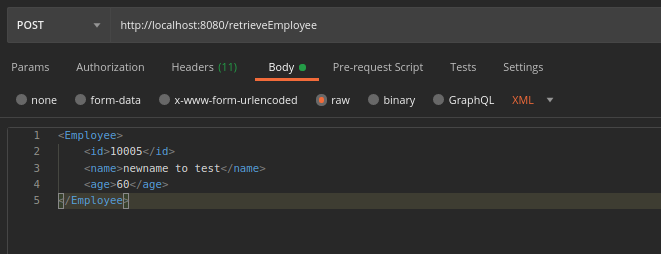
****

**Output:** [**http://localhost:8080/greet-internationalized/Akanksha-tyagi**](http://localhost:8080/greet-internationalized/Akanksha-tyagi) **(in message\_de.properties only)**

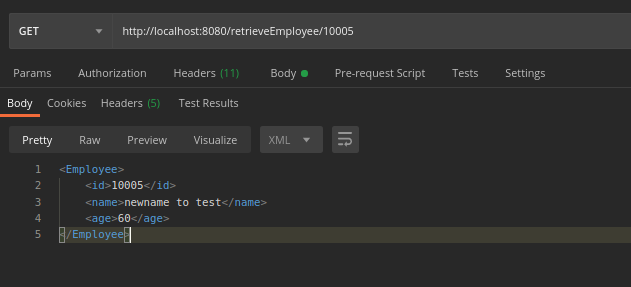
****

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

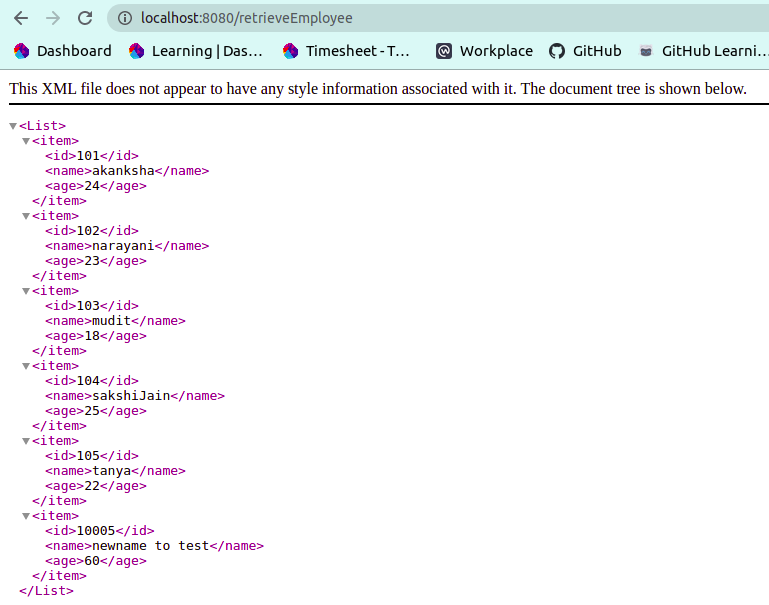
Adding dependencies of xml in pom.xml**Output:** creating employee by POST method:

[**http://localhost:8080/retrieveEmployee/10005**](http://localhost:8080/retrieveEmployee)

Getting the created employee

[**http://localhost:8080/retrieveEmployee**](http://localhost:8080/retrieveEmployee) ****

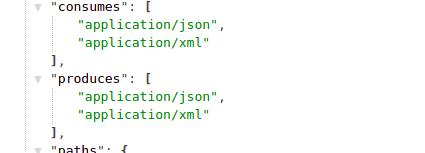
**4. Create GET Method to fetch the list of users in XML format.**

****

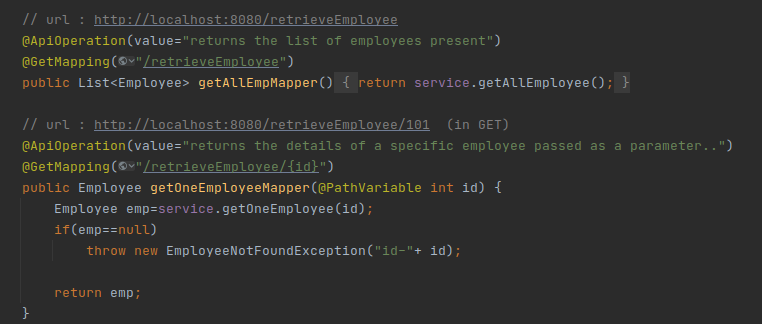
**5. Configure swagger plugin and create document of following methods:**

[**http://localhost:8080/v2/api-docs**](http://localhost:8080/v2/api-docs)

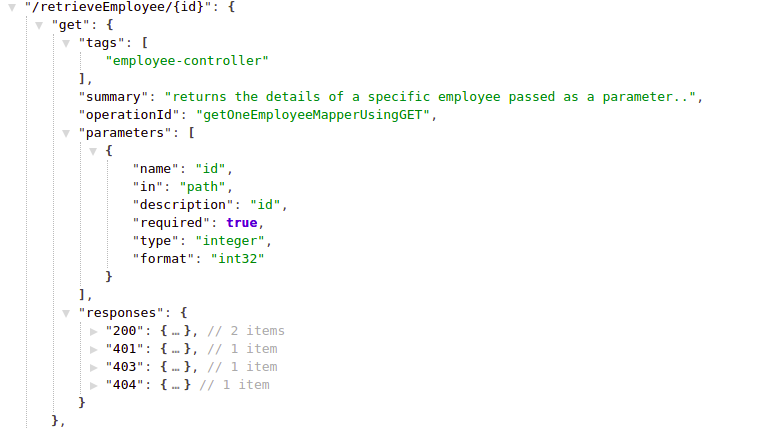
****

****

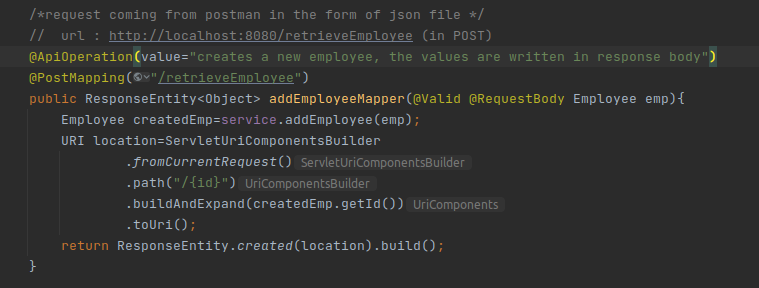
**Get details of User using a GET request.**

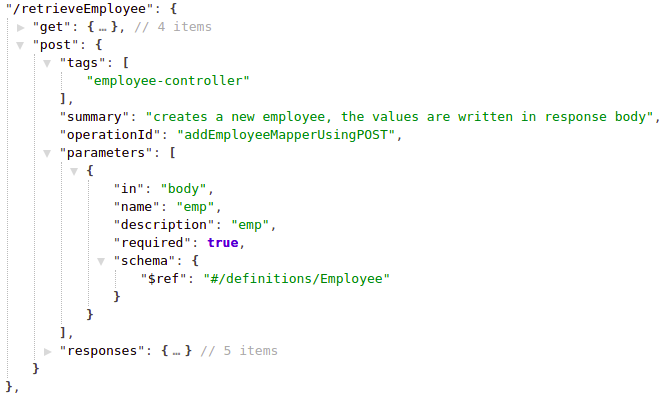
****

****

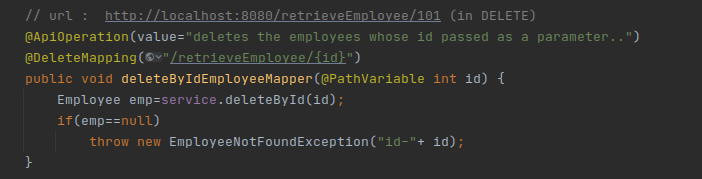
****

**Save details of the user using POST request.**

****

****

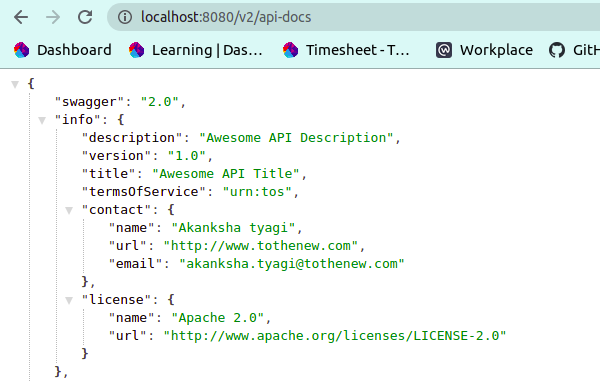
**Delete a user using DELETE request.**

****

****

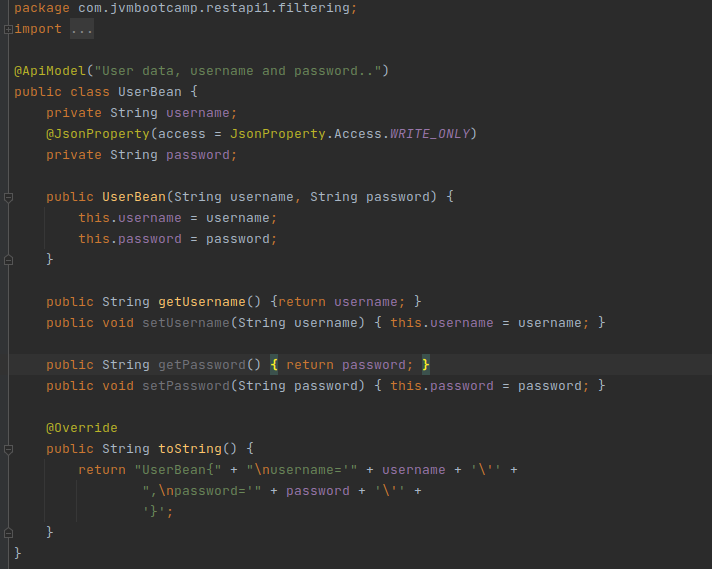
**6. In swagger documentation, add the description of each class and URI so that in swagger UI the purpose of class and URI is clear.**

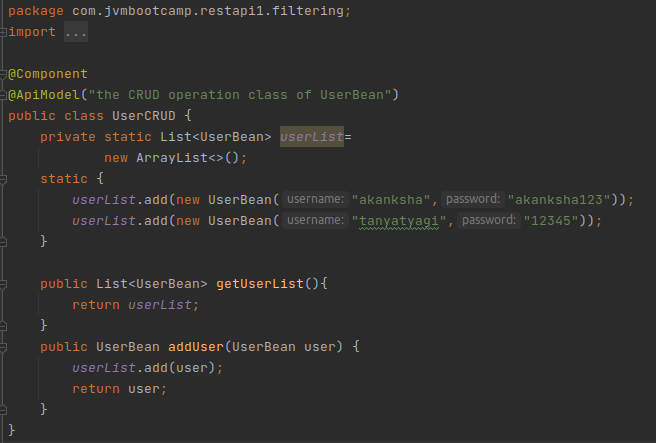
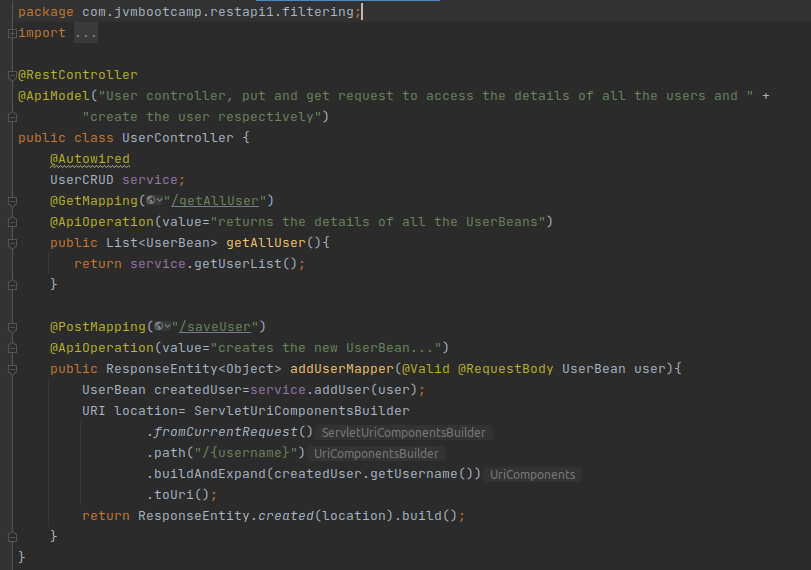
[**http://localhost:8080/v2/api-docs**](http://localhost:8080/v2/api-docs)

****

**class description :**

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

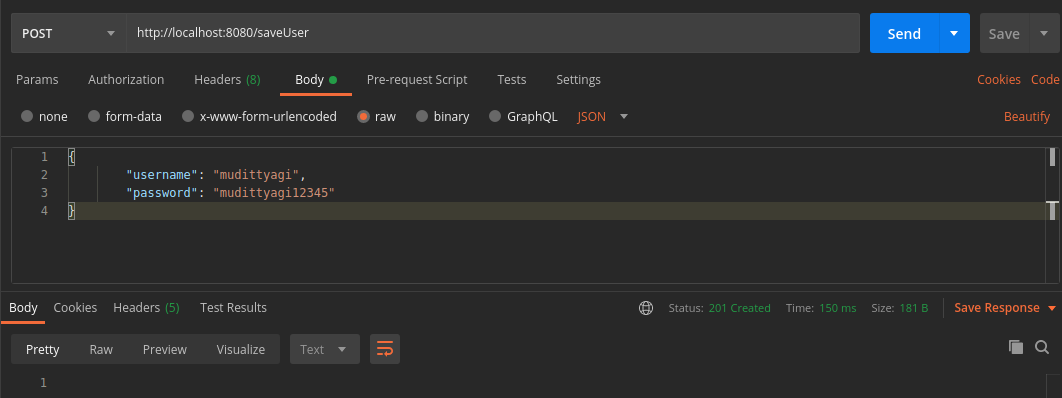
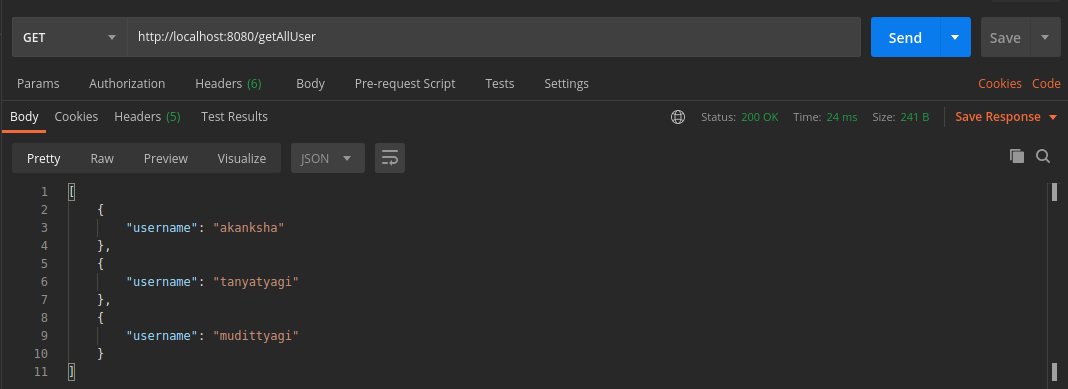
UserBean.java with properties username and password**:**

UserCRUD.java****Userntroller.java which accepts and responds to the incoming requests****

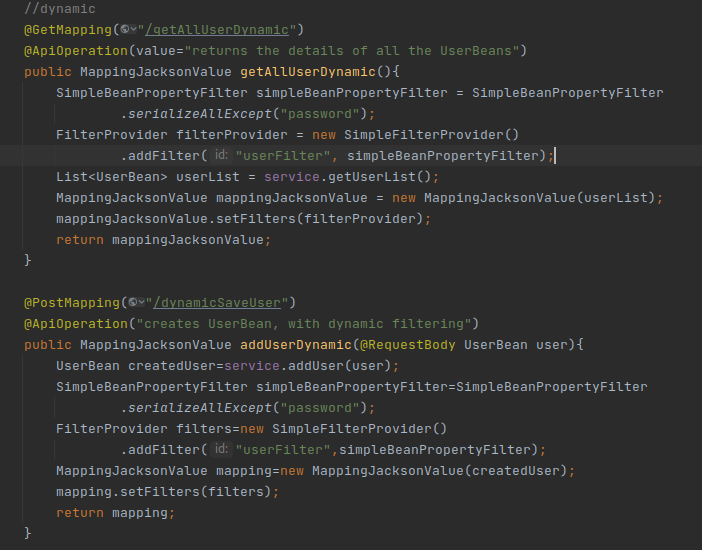
**Output:**

Creating user:

[**http://localhost:8080/saveUser**](http://localhost:8080/saveUser)

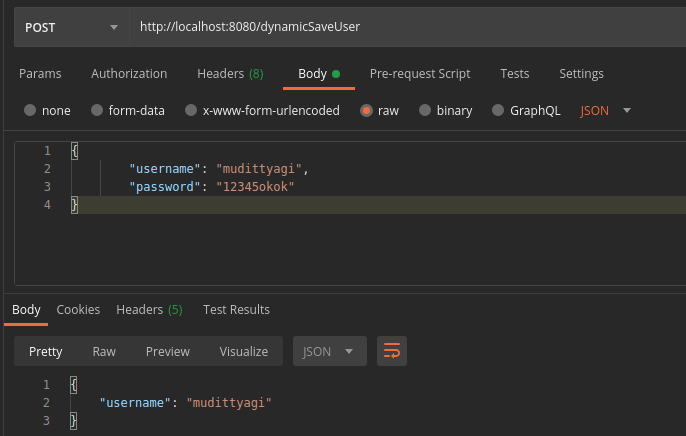
****displaying all the users: [**http://localhost:8080/getAllUser**](http://localhost:8080/getAllUser) ****

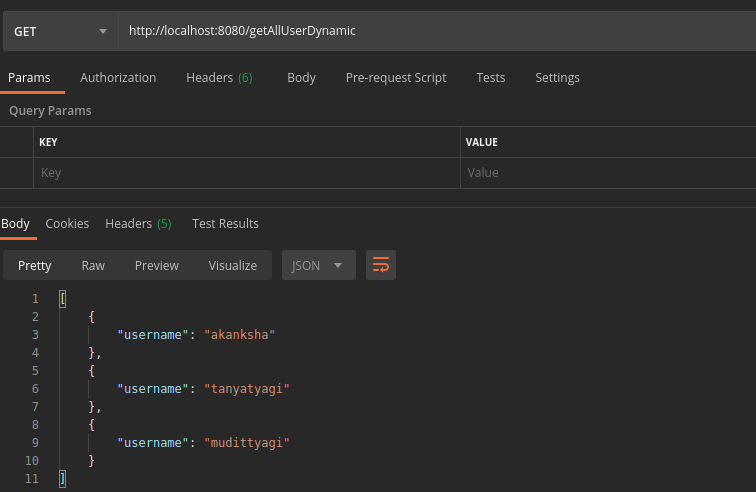
**8. Create another API that does the same by using Dynamic Filtering.**

****

**Output:**

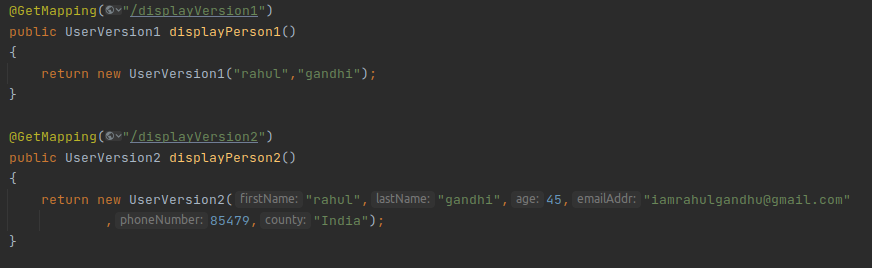
[**http://localhost:8080/dynamicSaveUser**](http://localhost:8080/dynamicSaveUser) **(POST)**

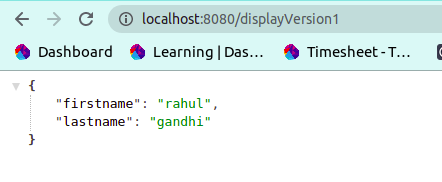
****

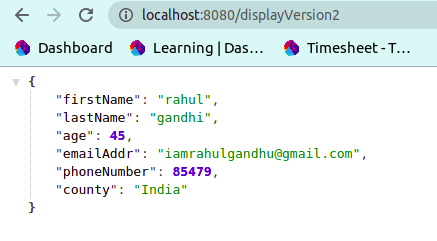
[**http://localhost:8080/getAllUserDynamic**](http://localhost:8080/getAllUserDynamic) ****

**9. 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,**

**Uri versioning:**

****

[**http://localhost:8080/displayVersion1**](http://localhost:8080/displayVersion1) ****

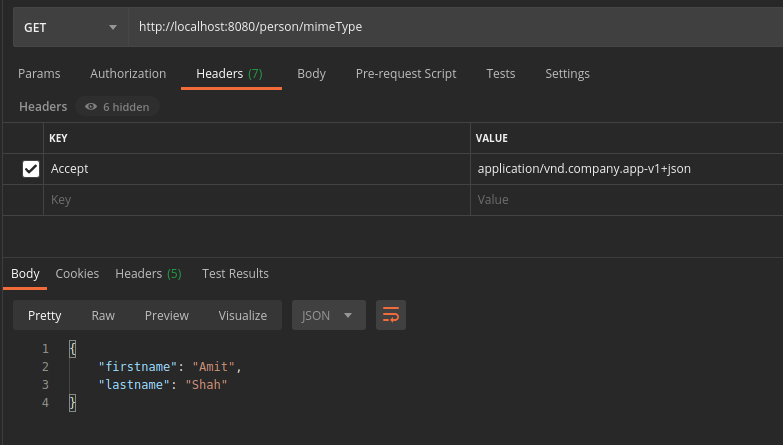
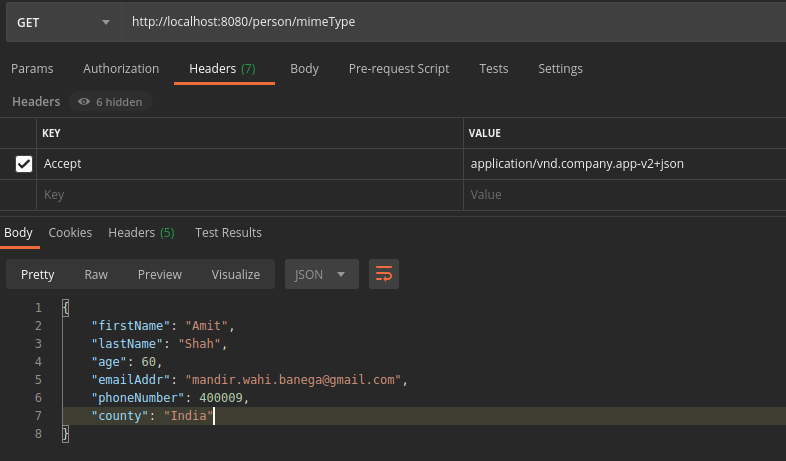
[**http://localhost:8080/displayVersion2**](http://localhost:8080/displayVersion2) ****

**Now apply versioning using the following methods:**

**MimeType Versioning:**

****

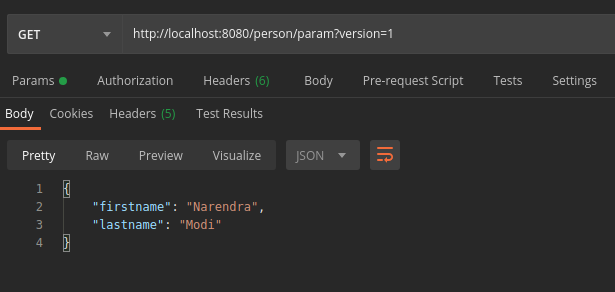
**Output:**

[**http://localhost:8080/person/mimeType**](http://localhost:8080/person/mimeType) **with Accept- application/vnd.company.app-v1+json**[**http://localhost:8080/person/mimeType**](http://localhost:8080/person/mimeType) **with Accept- application/vnd.company.app-v2+json**

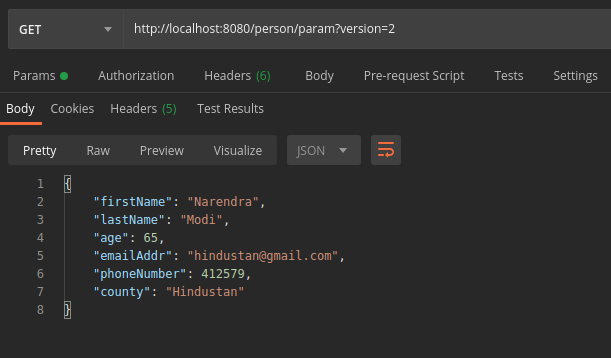
**Request Parameter versioning**

****

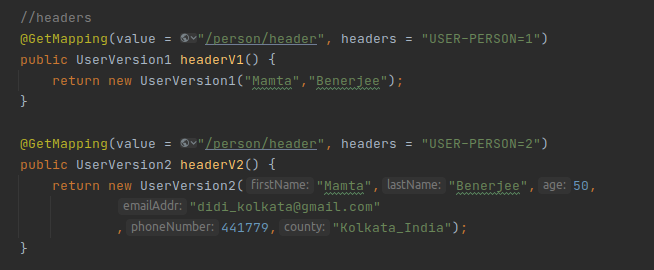
**Output:**

[**http://localhost:8080/person/param?version=1**](http://localhost:8080/person/param?version=1) ****

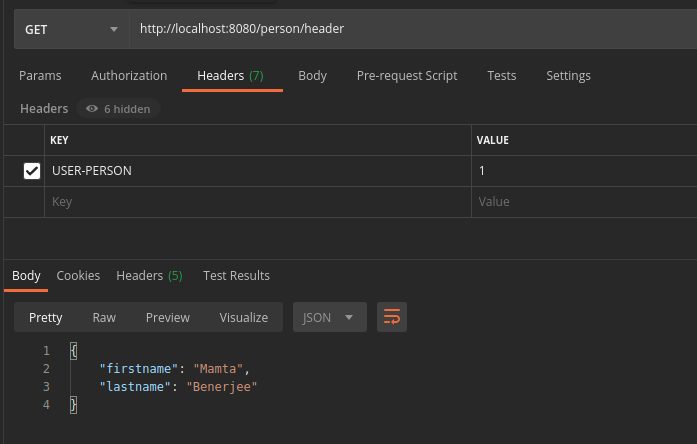
[**http://localhost:8080/person/param?version=2**](http://localhost:8080/person/param?version=2)

****

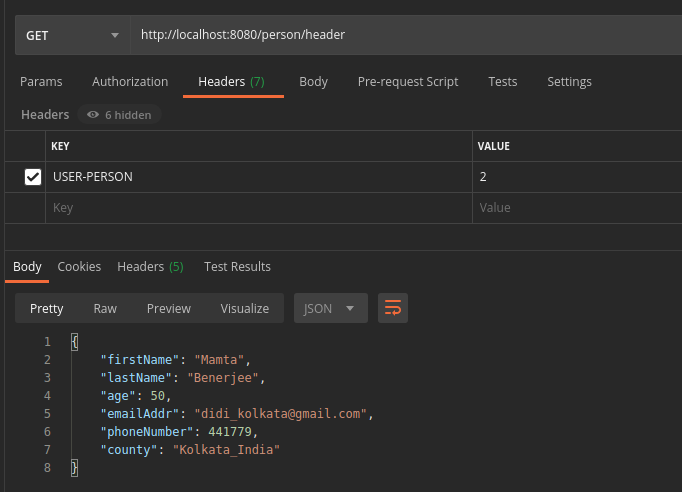
**Custom Header Versioning:**

****

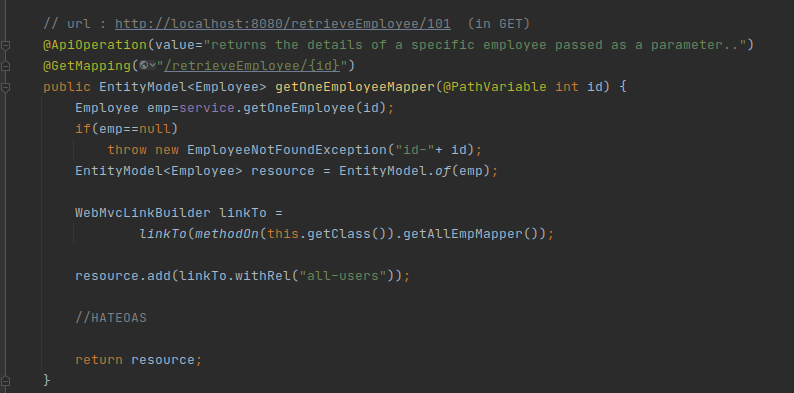
**Output:**

[**http://localhost:8080/person/header**](http://localhost:8080/person/header) **with USER-PERSON=1**

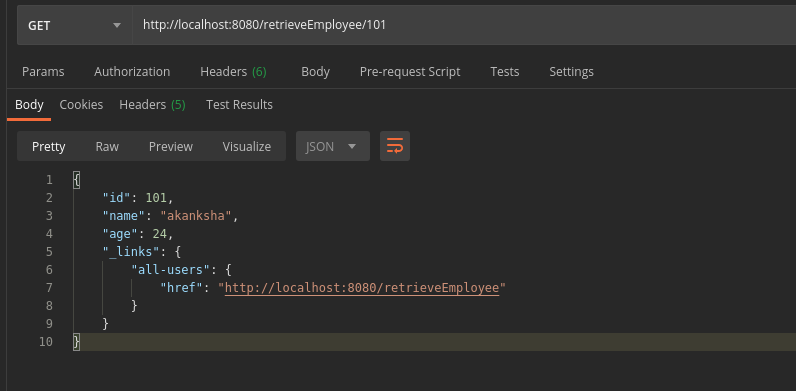
[**http://localhost:8080/person/header**](http://localhost:8080/person/header) **with USER-PERSON=2**

****

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

****

**Output:**

[**http://localhost:8080/retrieveEmployee/101**](http://localhost:8080/retrieveEmployee/101) ****