Internationalization

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

Ans == <https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/Hello_world_controller.java>

<https://github.com/pranav0297/restFull/tree/branch2/src/main/resources>

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

Ans == <https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/param.java>

\*Content Negotiation

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

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

Ans 3 and 4===== <https://github.com/pranav0297/restFull/blob/branch2/pom.xml>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/Employee1_controller.java>

\*Swagger

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

Get details of User using GET request.

Save details of the user using POST request.

Delete a user using DELETE request.

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.

Ans 5 and 7 === Ans <https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/swagger_config.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/Employee1_controller.java>

\*Static and Dynamic filtering

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

Ans ==

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/filltering_controller1.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/filter_example.java>

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

Ans == <https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/filltering_controller.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/filter_example.java>

\*Versioning Restful APIs

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

Ans == <https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/versioning_controller.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/versioning_person1.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/versioning_person2.java>

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/Name.java>

\*HATEOAS

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

Ans == @GetMapping("/Employee-one/{id}")

public EntityModel<Employee1>employeeList2(@PathVariable int id){

Employee1 e= Employe\_dao.*findOne*(id);

if(e== null)

throw new UserNotFoundException("id-"+id);

EntityModel<Employee1> resourse = EntityModel.*of*(e);

WebMvcLinkBuilder linkTo =*linkTo*(*methodOn*(this.getClass()).employeeList2(id));

resourse.add(linkTo.withRel("all-user"));

return resourse;

}

<https://github.com/pranav0297/restFull/blob/branch2/src/main/java/com/restFull_services/resFull_demo/Employee1_controller.java>

<https://github.com/pranav0297/restFull/blob/branch2/pom.xml>