CMP SCI 4012/5012 – Project 3

Creating a RESTful API:

User Interface

- There is no UI for this project.

Features

Path	Operation	Request Body	Description
/project3/user/{user}	OPTIONS	Empty	Discovery of all operations for /project/user/{userId}
/project3/user/{userId}	GET	Empty	This pulls back data on a user based off given userId.

/project3/user/{userId}	PUT	{
		“username”:”value1”,
		“password”:”value2”,
		“firstname”:”value3”,
		“lastname”:”value4”
		}
			Updates the user.

/project3/user/{userId}	DELETE	{
		“adminusername”:”admin”,
		“adminpassword”:”admin”
		}
			Deletes the user. Only allow admins to do this.

/project3/user	POST	{
		“username”:”value1”,
		“password”:”value2”,
		“firstname”:”value3”,
		“lastname”:”value4”
		}
			Registers a new user.

/project3/user	OPTIONS	Empty	Discovery of all operations for /project/user
/project3/user	GET	Empty	Returns a list of all users.

- Return 200 for successful GET requests and response body contains JSON of requested object.
- Return 201 for successful POST request and response body contains JSON of the created object. The response header should include Location header with the URL for the newly created resource. Required data fields are username, password, firstname, and lastname to create a new user. Return a 400 error if not all data fields are set and indicate which fields need to be set in the response body.

- Return 204 for successful PUT request and empty response body. No required fields for a PUT request, the request body can contain username, password, firstname, and lastname. All are optional. No error is thrown if there is a PUT request with an empty request body.
- Return 204 for successful DELETE request and empty response body. A second request to DELETE the same object returns a 404 error.
- Return a 403 error whenever someone other than the admin tries to DELETE a user.
    - The admin username should be "admin" and password should be "admin", this way I can test.
- Return 204 for successful OPTION requests with Allow header whose contents are a comma-separated list of the HTTP methods supported for the resource.

Database

- Exact database implementation is up to each individual. Since we have not covered Hibernate yet, please use JDBC for a database connection to MySQL. I do have some high level suggestions for the database.
- Create MySQL database called "enterprise"
- Create USERS table
    - USERS_ID column, auto increment
    - USERNAME column, varchar(whatever size you wish)
    - PASSWORD column, varchar(whatever size you wish)
    - FIRSTNAME column, varchar(whatever size you wish)
    - LASTNAME column, varchar(whatever size you wish)
    - LASTUPDATED date

TECHNOLOGY STACK

- Hibernate
- JSON
- SPRING MVC (Root Context and then Servlet Contexts)
- Follow best practices