CS 230 Module Four Assignment Guidelines and Rubric

Overview

Your client, The Gaming Room, currently has a game application, Draw It or Lose It, that was created for Android systems. They would like to expand it to other operating platforms. To do so, you have been asked to create a simulated environment that demonstrates prototypes to present multiple platform options to your client.

In this assignment, you **will develop components of an application so it may be deployed on various operating platforms** using the client-server architectural pattern. You will use Basic Authentication to build a username and password login interface by modifying the provided tutorial code. Primarily, you will be working on the server-side component of the application to gain experience with developing using a REST Application Programming Interface (API).

Effective interfaces and security features are important to your client, so you will need to address this in your prototype. While Basic Authentication is not the most robust and secure mechanism, it will take you through all the steps of preparing your application to incorporate security and thus be protected against unauthorized access. Later, you can replace the specific authentication mechanism with a more robust version and your application will be ready to support that.

Prompt

Review the Dropwizard BasicAuth Security Example and the Dropwizard Jersey/HTTP Configuration and Examples page to support your development of the code. These examples build on the beginner "Hello World" example by showing how to add basic authentication to your RESTful API.

Begin by importing the gameauth.zip file into Eclipse as a new Maven project. If you have not already practiced using Maven, be sure to follow the instructions in the Dropwizard Hello World Tutorial PDF.

Note: You may need to update the dropwizard version in the pom.xml file to align with the version that you are using.

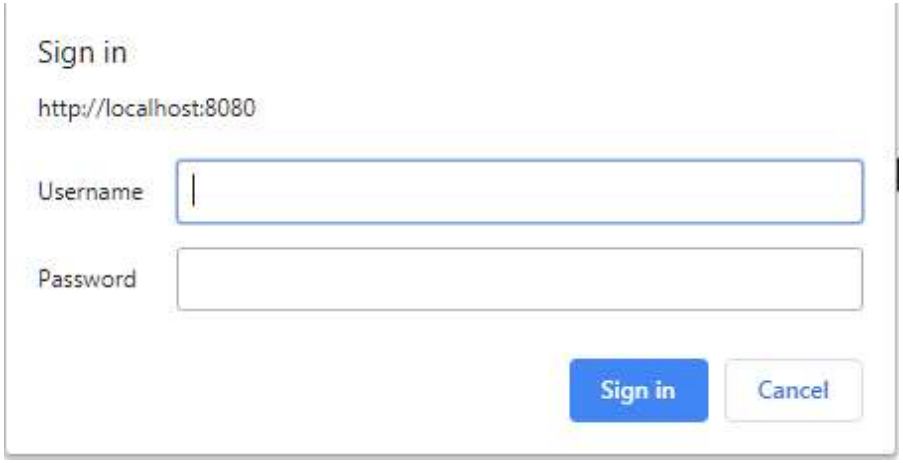
To view the specific tasks you have been assigned, navigate in your Eclipse project to the **Window**, then **Show View**, then **Tasks** view. You can double-click on any of the tasks to be directed to the line of code where the task that needs to be completed is located. Specifically, review and complete all assigned tasks outlined below as indicated in the FIXME comments within the various Java files. A text version of this image is available: FIXME Text Version Word Document.

- **User Interface: Write code to create the user interface for entering data for user authentication and securing REST APIs with the proper annotation**. To do this, you must complete the following tasks in the GameAuthApplication.java file based on the BasicAuth Security Example:
  - Register the GameUserREST Controller.
  - Create a JerseyClientBuilder instance named "DemoRESTClient".
- **Client: Write code to expose the Controller class and create a path for the client-side endpoint**. To do this, you must complete the following tasks in the corresponding Java files:
  - GameUserRESTController.java: Add the proper annotation to expose this Controller class with the base endpoint URL as /gameusers.

- RESTClientController.java: Add the proper annotation for the HTTP GET method and a path of **/gameusers/** to expose an endpoint in order to return a list of all game users.

- **Authentication and Authorization (Server): Create a username and password-based authentication and role-based authorization for users, using annotations that set up permissions for the users**. To do this, you must complete the following tasks in the corresponding Java files based on the BasicAuth Security Example:
  - GameAuthenticator.java: Complete the Authenticator method by verifying a new instance of the GameUser class based upon the username that has been authenticated.
  - GameAuthorizer.java: Complete the Authorizer method by roles of the user in the GameUser class based upon the username that has been authenticated. This completes the interface requirement for authentication using the Basic Authentication method.

- **User Protection and Security: Complete a REST Application including all classes, resources, and representations, along with APIs that request validation and are tested and verified using data provided**. To do this, you must complete the following tasks in the GameUserRESTController.java file based on the BasicAuth Security Example:
  - Add the proper RolesAllowed annotation to restrict the createGameUser HTTP POST method to only authenticated users with the ADMIN role.
  - Add the proper RolesAllowed annotation to restrict the getGameUserById HTTP GET method to only authenticated users with the USER role.
  - Test your code by running the application in Eclipse.
    - Remember to add the values "server config.yml" as Program Arguments on the **Arguments** tab of **Run Configuration**.
    - Use any web browser to visit the URL **http://localhost:8080/gameusers**. The browser's Basic Authentication dialog should appear with blank fields for Username and Password as shown below:



    - Valid usernames to try are "guest", "user", "player", and "admin", all with the same password value of "password". Upon successful authentication, you will see the list of game players returned as a JSON string:

[{"id":1,"firstName":"Lokesh","lastName":"Gupta","email":"India"},
{"id":2,"firstName":"John","lastName":"Gruber","email":"USA"},
{"id":3,"firstName":"Melcum","lastName":"Marshal","email":"AUS"}]

Visiting the URL **http://localhost:8080/gameusers/1** while logged in as a user in the USER role will return the player whose ID is 1 (Lokesh Gupta). Visiting the URL with **/2** will return the player whose ID is 2 (John Gruber), and so on. Close the browser, then reopen it and try accessing the same URL as a player or guest, and you should see the following JSON string displayed:

```
{"code":403,"message":"User not authorized."}
```

- **Industry Standard Best Practices**: Be sure to demonstrate industry standard best practices in your code, including descriptive in-line comments and appropriate variable and method naming conventions.

What to Submit

Game Player Management Application

Submit the completed application code. Be sure to download and save the project. Compress your Eclipse project directory into a single ZIP file to be submitted.

Module Four Assignment Rubric

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
User Interface	Writes user interface code for entering data and user authentication, naming and registering client	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include use of custom classes or proper configuration of custom classes	Does not attempt criterion	25
Client	Writes code to expose controller class and create path for client-side endpoint	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include exposing the controller class and creating a path for client	Does not attempt criterion	20
Authentication and Authorization (Server)	Creates username and password-based authentication and role-based authorization for users, using annotations that set up permissions for the users	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include the inclusion of authentication or authorization	Does not attempt criterion	20
User Protection and Security	Completes a REST Application including all classes, resources, and representations, along with APIs that request validation and are tested and verified using data provided	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include the inclusion of all components in the REST Application Class	Does not attempt criterion	25

Criteria	Proficient (100%)	Needs Improvement (70%)	Not Evident (0%)	Value
Industry Standard Best Practices	Demonstrates industry standard best practices including appropriate in-line comments and variable and method naming conventions	Shows progress toward proficiency, but with errors or omissions	Does not attempt criterion	10
			Total:	100%