Documentation of the code LoginServletProject

LoginServlet.java

```
1. import java.io.IOException;
2. import javax.servlet.ServletException;
3. import javax.servlet.annotation.WebServlet;
4. import javax.servlet.http.HttpServlet;
5. import javax.servlet.http.HttpServletRequest;
6. import javax.servlet.http.HttpServletResponse;

// These lines import necessary Java classes and servlet packages for handling HTTP requests and responses.
// They provide functionalities for handling exceptions, servlets, and HTTP communication.
8. @WebServlet("/LoginServlet")
// This annotation declares that the following class is a servlet and maps it to the specified URL pattern "/LoginServlet".
9. public class LoginServlet extends HttpServlet {
// This line declares a class named LoginServlet which extends HttpServlet, indicating that it's a servlet.
10. private static final long serialVersionUID = 1L;
// This line declares a private static final long variable serialVersionUID to ensure version control compatibility.
12. protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
// This line starts the definition of a method named doPost which handles HTTP POST requests.
13. // Hardcoded username and password for validation
// This comment explains that the username and password for validation are hardcoded in the servlet code.
14. String validUsername = "aayush";
15. String validPassword = "pwd2326";
// These lines declare and initialize variables to store the valid username and password for validation.
17. // Get the username and password entered by the user
// This comment explains that the servlet is retrieving the username and password entered by the user.
18. String username = request.getParameter("username");
19. String password = request.getParameter("password");
// These lines retrieve the username and password parameters from the HTTP request sent by the client.
21. // Check if the entered username and password match the valid credentials
// This comment explains that the servlet is comparing the entered username and password with the valid credentials.
22. if (username != null && password != null && username.equals(validUsername) && password.equals(validPassword)) {
// This line checks if both the username and password are not null and if they match the valid credentials.
23. // Redirect to redirect.html with username as parameter
// This comment explains that if the credentials are correct, the servlet will redirect the user to "redirect.html" with the username as a parameter.
24. response.sendRedirect("redirect.html?username=" + username);
// This line redirects the user to "redirect.html" with the username as a parameter in the URL.
25. } else {
// If the entered credentials are incorrect, execute the following block of code.
26. // Credentials are incorrect, redirect back to login.html
// This comment explains that if the entered credentials are incorrect, the servlet will redirect the user back to "login.html".
27. // Notify user to give correct username and password
// This comment explains that the user will be notified to provide the correct username and password.
```

28. response.sendRedirect("login.html?error=incorrect");
// This line redirects the user back to "login.html" with an error parameter indicating incorrect credentials.
30. }
// End of the if-else block.
32. protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
// This line starts the definition of a method named doGet which handles HTTP GET requests.
33. // Redirect to login.html for GET requests
// This comment explains that for GET requests, the servlet will redirect the user to "login.html".
34. response.sendRedirect("login.html");// This line redirects the user to "login.html" for HTTP GET requests.
35. }// End of the doGet method definition.
36. }// End of the class definition.

Login.html

1. <!DOCTYPE html>
// This line declares the document type and version of the HTML document.
2. <html lang="en">
// This line starts the HTML document and specifies the language as English.
3. <head>
// This line starts the head section of the HTML document, which contains meta-information about the document.
4. <meta charset="UTF-8">
// This line specifies the character encoding of the document as UTF-8, which supports a wide range of characters.
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
// This line sets the viewport properties for responsive web design, ensuring that the page content scales properly on different devices.
6. <title>Login Page</title>
// This line sets the title of the HTML document as "Login Page".
7. </head>
// End of the head section.
8. <body>
// This line starts the body section of the HTML document, which contains the visible content of the page.
9. <h1>This is the login page</h1>
// This line creates a level 1 heading displaying the text "This is the login page".
10. <form action="LoginServlet" method="post">
// This line starts a form element with the action attribute set to "LoginServlet" and the method attribute set to "post".
11. <label for="username">Username:</label>
// This line creates a label for the username input field.
**12. <input type="text" id="username" name="username" required>

**
// This line creates a text input field for the username with the id "username", name "username", and the required attribute to ensure it's filled out.
13. <label for="password">Password:</label>
// This line creates a label for the password input field.
**14. <input type="password" id="password" name="password" required>

**
// This line creates a password input field for the password with the id "password", name "password", and the required attribute for validation.
15. <input type="submit" value="Login">
// This line creates a submit button with the text "Login" to submit the form data.
16. </form>

// End of the form element.
17. </body>
// End of the body section.
18. </html>
// End of the HTML document.

Redirect.html

1. <!DOCTYPE html>
// This line declares the document type and version of the HTML document.
2. <html lang="en">
// This line starts the HTML document and specifies the language as English.
3. <head>
// This line starts the head section of the HTML document, which contains meta-information about the document.
4. <meta charset="UTF-8">
// This line specifies the character encoding of the document as UTF-8, which supports a wide range of characters.
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
// This line sets the viewport properties for responsive web design, ensuring that the page content scales properly on different devices.
6. <title>Successful Login</title>
// This line sets the title of the HTML document as "Successful Login".
7. </head>
// End of the head section.
8. <body>
// This line starts the body section of the HTML document, which contains the visible content of the page.
9. <h1>Successful Login</h1>
// This line creates a level 1 heading displaying the text "Successful Login".
**10. Hello , welcome! **
// This line creates a paragraph element with a span element inside it, which will be used to display the username dynamically.
12. <script>
// This line starts a script block where JavaScript code is placed.
13. // Function to retrieve URL parameter by name
// This comment explains that the following function retrieves a parameter from the URL by name.
14. function getUrlParameter(name) {
// This line declares a JavaScript function named getUrlParameter which takes a parameter 'name'.
15. name = name.replace(/[\[]/, '\\[').replace(/[\]]/, '\\]');
// This line replaces square brackets in the parameter 'name' with escape characters to ensure proper matching.
16. var regex = new RegExp('[\\?&]' + name + '=([^&#]*)');
// This line constructs a regular expression to match the parameter 'name' in the URL.
17. var results = regex.exec(location.search);
// This line executes the regular expression on the URL's search portion to retrieve the parameter value.
18. return results === null ? '' : decodeURIComponent(results[1].replace(/\+/g, ' '));
// This line returns the decoded parameter value or an empty string if it's not found.
19. }
// End of the getUrlParameter function definition.
21. // Get the username parameter from the URL
// This comment explains that the username parameter is retrieved from the URL.
22. var username = getUrlParameter('username');

// This line calls the getUrlParameter function to retrieve the value of the 'username' parameter from the URL.

24. // Display the username on the page

// This comment explains that the retrieved username will be displayed on the page.

25. document.getElementById('username').textContent = username;

// This line sets the text content of the span element with the id 'username' to the retrieved username.

26. </script>

// End of the JavaScript code.

27. </body>

// End of the body section.

28. </html>

// End of the HTML document.