```java
package fi.paatti.paattiloginutils;


import com.vaadin.data.Container;
import com.vaadin.data.Item;
import com.vaadin.data.util.sqlcontainer.SQLContainer;
import fi.paatti.paattidatabaseutils.dbobjects.User;
import fi.paatti.paattidatabaseutils.dbobjects.UserGroup;
import fi.paatti.paattidatabaseutils.dbservice.PaattiDBService;
import fi.paatti.paattidatabaseutils.names.PaattiColumnNames;
import fi.paatti.paattidatabaseutils.names.PaattiTableNames;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * The helper class for user login. It handles the login and and controls the
 * cookies. It creates an User object that has users groups, roles and
 * researches.
 *
 * @author Toni Salminen, toni.a.j.salminen@student.jyu.fi
 * @author Lauri Satokangas, lauri.n.satokangas@student.jyu.fi
 */
public class LoginHelper {

  private PaattiDBService dbService;
  private HttpServletRequest request;
  private HttpServletResponse response;

  /**
   * The constructor.
   *
   * @param db the database service to be used.
   * @param request the HttpServletRequest.
   * @param response the HttpServletResponse.
   */
  public LoginHelper(PaattiDBService db, HttpServletRequest request, HttpServletResponse response) {
    this.dbService = db;
    this.request = request;
    this.response = response;
  }

  /**
   * Sets the HttpServletRequest.
   *
```

```
   * @param request the HttpServletRequest to be set.
   */
  public void setRequest(HttpServletRequest request) {
    this.request = request;
  }

  /**
   * Sets the HttpServletResponse.
   *
   * @param response the HttpServletResponse to be set.
   */
  public void setResponse(HttpServletResponse response) {
    this.response = response;
  }

  /**
   * Returns the User object using the given username and password.
   *
   * @param username the user's name.
   * @param password the user's password.
   * @return The User object if successful or null if not successful.
   */
  public User getUser(String username, String password) {

    boolean succesful = false;
    SQLContainer container = dbService.getSQLContainerFromDBTable(PaattiTableNames.USER, PaattiColumnNames.USER_userID,
        true);
    Item item;
    Integer userID = null;

    for (Object id : container.getItemIds()) {
      item = container.getItem(id);
      if (item.getItemProperty(PaattiColumnNames.USER_name).getValue().toString().equals(username)
          && item.getItemProperty(PaattiColumnNames.USER_password).getValue().toString().equals(password)) {
        userID = (Integer) item.getItemProperty(PaattiColumnNames.USER_userID).getValue();
        succesful = true;
        break;
      }
    }
    if (succesful) {
      User user = new User(userID);

      user.setUsername(username);
      setUserGroupsRolesAndResearches(user, dbService.getUserGroupsdRolesAndResearches(userID));
      return user;
    } else {
```

```
      return null;
    }
  }

  /**
   * Tries to log in with the browser cookies.
   *
   * @param usernameCookieName the name of the cookie containing the user name.
   * @param passwordCookieName the name of the cookie containing the user password.
   * @return The user matching the given (cookie) values, otherwise null.
   */
  public User TryLoginWithCookies(String usernameCookieName, String passwordCookieName) {
    String username = getCookieValue(usernameCookieName);
    String password = getCookieValue(passwordCookieName);

    if (username != null && password != null) {
      // Cookies found. Try login.
      return getUser(username, password);
    }
    // No cookies found. Login failed.
    return null;
  }

  /**
   * Sets the login cookies in the browser memory.
   *
   * @param usernameCookieName the user name cookie name.
   * @param username the user name.
   * @param passwordCookieName the password cookie name.
   * @param cookieLifetime the lifetime of the cookie.
   * @param password the user's password.
   * @param cookiePath the path to cookie.
   */
  public void setLoginCookie(String usernameCookieName, String username,
      String passwordCookieName, String password, String cookiePath,
      int cookieLifetime) {

    setCookie(usernameCookieName, username, cookiePath, cookieLifetime);
    setCookie(passwordCookieName, password, cookiePath, cookieLifetime);
  }

  /**
   * Removes the specified login cookie from the browser memory.
   *
   * @param usernameCookieName the user name cookie name.
   * @param passwordCookieName the password cookie name.
```

```
   * @param cookiePath the path for the cookie.
   */
  public void deleteLoginCookie(String usernameCookieName,
      String passwordCookieName, String cookiePath) {

    int cookieLifetime = 0;

    setCookie(usernameCookieName, "", cookiePath, cookieLifetime);
    setCookie(passwordCookieName, "", cookiePath, cookieLifetime);
  }

  /**
   * Sets the cookie in browser memory.
   *
   * @param cookieName the name for the cookie.
   * @param cookieValue the value for the cookie.
   */
  private void setCookie(String cookieName, String cookieValue,
      String cookiePath, int cookieLifetime) {

    Cookie cookie = new Cookie(cookieName, cookieValue);

    cookie.setPath(cookiePath);
    cookie.setMaxAge(cookieLifetime);
    response.addCookie(cookie);
  }

  /**
   * Returns the cookies value from browser memory.
   * <p/>
   * @param cookieName The name of the cookie
   * @return The cookie
   */
  private String getCookieValue(String cookieName) {
    String cookieValue = null;

    Cookie[] cookies = request.getCookies();

    if (cookies == null) {
      return cookieValue;
    }

    for (int i = 0; i < cookies.length; i++) {
      Cookie cookie = cookies[i];

      if (cookie.getName().equals(cookieName)) {
```

```
        cookieValue = cookie.getValue();
      }
    }

    return cookieValue;
  }

  /**
   * Gets the users groups, roles and researches from the container and sets
   * them to the User-object.
   * <p/>
   * @param user The user object
   * @param ugContainer The container containing the user's groups
   */
  private void setUserGroupsRolesAndResearches(User user, Container.Indexed ugContainer) {
    Object ugItemId = ugContainer.firstItemId();
    Item ugItem;

    while (ugItemId != null) {
      ugItem = ugContainer.getItem(ugItemId);

      int usergroupID = (Integer) ugItem.getItemProperty(PaattiColumnNames.USERGROUP_usergroupID).getValue();
      int roleID = (Integer) ugItem.getItemProperty(PaattiColumnNames.ROLE_roleID).getValue();
      int researchID = (Integer) ugItem.getItemProperty(PaattiColumnNames.RESEARCH_researchID).getValue();

      UserGroup ugGroup = new UserGroup(usergroupID, roleID);

      ugGroup.setUsergroupResearchID(researchID);
      user.addUserGroup(ugGroup);

      ugItemId = ugContainer.nextItemId(ugItemId);
    }
  }
}
```