

Table of Contents

[Table of Contents](#)

[Project objective](#)

[Background of the problem statement](#)

[Specifications](#)

[Java Concepts and Data Structures](#)

[Repository](#)

[Flow of the application](#)

[Screenshots](#)

[Main Menu \(Index page\)](#)

[View Users page](#)

[Find User Page](#)

[Find User Result: User Found](#)

[Find User Result: User Not Found](#)

[Add New User Page](#)

[Add New User Success](#)

[Edit User Page](#)

[Edit User Success](#)

[Login page](#)

[Login Success](#)

[Login Failed](#)

USER WEB APPLICATION

Developer: Diemi Pham - HCL America Inc.

Date: December, 2020

Project objective

- **Retrieving the User Details Using the User ID:** Create a servlet-based application that shows a form to enter a user ID. The user ID is then validated, and user details are retrieved from the database and displayed to the user. You need to create a user table in H2 and pre-populate it with data. Use JDBC to do all database processing.
- **Adding a New User in the Database:** Create a servlet-based web application that shows a form to add new users. A H2 database will be created to store user data. The form data will be validated, and a row will be added to the database. All database processing will be done using Hibernate.
- **User Details Portal:** Create a servlet-based web application that shows a form to enter user details. Capture the details in a servlet and then display the data that was entered.
- **Validation of the User Login:** Create a servlet-based web application that shows a login page and validates it. A H2 database will be created to store user data. On successful login, a dashboard page is shown. The dashboard will provide a link for logging out. Incorrect logins need to be handled by showing an error message page.

Background of the problem statement

- **Retrieving the User Details Using the User ID:** As a part of developing an e-commerce web application, the admin backend requires a module that can retrieve user information based on the user ID.
- **Adding a New User in the Database:** As a part of developing an e-commerce web application, you have to create a database table for storing user information. A form is needed to add new users. The form submission is validated, and a new record is created in the user table.
- **User Details Portal:** As a part of developing an ecommerce web application, you have to prototype a form for adding users into the system entered by the users. There is no database involved here, so the product is just captured and displayed without storing it anywhere. (Notes: this requirement is upgraded to store entered information into the database. Hence, this becomes the feature *Adding a New User in the Database* mentioned above.)
- **Validation of the User Login:** As a part of developing an e-commerce web application, you have to prototype a login scenario for the user. A form is needed for users to enter username and password. The form submission is validated, and users are redirected to a welcome page.

Specifications

- **Retrieving the User Details Using the User ID:**
 - There is a JSP page to take in a user ID
 - A servlet that will take the user ID and use JDBC to query the database for the user
 - If the user is found, the servlet will display the user details, otherwise it will show an error message
- **Adding a New User in the Database:**
 - Show an add user form in JSP
 - Handle the form submission in a servlet. Validate it for any missing information
 - Add the user information into the database using Hibernate
 - A Hibernate configuration file is set up using XML for the user table
 - A class is created to store user data. This class will be linked to Hibernate via an hbm.xml file
 - The servlet will put the form data into the user class and then pass the user class into Hibernate to add into the database
- **User Details Portal:** is upgraded to *Adding a New User in the Database*
- **Validation of the User Login:**
 - Show a login form in HTML
 - Handle invalid logins and show appropriate error messages using servlets
 - Show the dashboard page using servlets
 - Handle logouts using servlets

Java Concepts and Data Structures

- H2 database is used for storing user data;
- Servlet and JDBC are used to query the database for the user.
- Hibernate is used for database processing

Repository

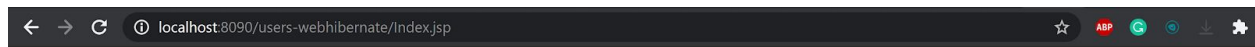
[Source Codes](#)

[Technical Documentation](#) (online version)

Flow of the application

Screenshots

Main Menu (Index page)



Menu

IMPORTANT: Please generate sample users or add at least one user to test LOGIN functionality.

[Login](#)

[Add New User](#)

[Find User](#)

[View Users](#)

[Generate Sample Users](#) (Only run once. Otherwise, the program may not work correctly)

View Users page



User List

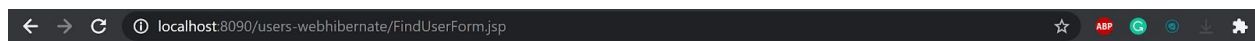
[Delete All Users](#) (WARNING: This action will delete all users from the database)

User ID	First Name	Last Name	Email	City	Edit	Delete
1	Mimi	Pham	mimi.pham@mail.com	San Antonio	Edit	Delete
2	Firstname 2	Lastname 2	Firstname2.Lastname2@mail.com	San Antonio	Edit	Delete
3	Firstname 3	Lastname 3	Firstname3.Lastname3@mail.com	San Antonio	Edit	Delete
4	Firstname 4	Lastname 4	Firstname4.Lastname4@mail.com	San Antonio	Edit	Delete
5	Firstname 5	Lastname 5	Firstname5.Lastname5@mail.com	San Antonio	Edit	Delete
6	Diemi	Pham	diemi.pham@mail.com	Austin	Edit	Delete

[Add New User](#)

[Back to Menu](#)

Find User Page



Find User

User ID:

Find User Result - User Found

← → ↻ ⓘ localhost:8090/users-webhibernate/FindUser.jsp ☆ ABP ⚙️

Find User

User ID	First Name	Last Name	Email	City	Edit	Delete
1	Mimi	Pham	mimi.pham@mail.com	San Antonio	Edit	Delete

[Find Another User](#)
[Back to Menu](#)

Find User Result - User Not Found

← → ↻ ⓘ localhost:8090/users-webhibernate/UserNotFound.jsp ☆ ABP ⚙️

User Not Found

Cannot find any user with user ID you have entered. This user may not exist.

[Find User](#)
[Back to Menu](#)

Add New User Page

← → ↻ ⓘ localhost:8090/users-webhibernate/AddUserForm.jsp ☆ ABP ⚙️

Add New User

First Name:

Last Name:

Email:

City:

Add New User Success

← → ↻ ⓘ localhost:8090/users-webhibernate/AddUserSuccess.jsp ☆ ABP ⚙️

Add New User

Welcome Alex Pham

[Add New User](#)
[Back to Menu](#)
[Logout](#)

Edit User Page

← → ↻ ⓘ localhost:8090/users-webhibernate/EditUserForm.jsp?id=7 ☆ ABP 🔍 ⬇ ⚙

Edit Form

User ID cannot be changed.

User ID:

First Name:

Last Name:

Email:

City:

Edit User Success

← → ↻ ⓘ localhost:8090/users-webhibernate/ViewUsers.jsp ☆ ABP 🔍 ⬇ ⚙

User List

[Delete All Users](#) (WARNING: This action will delete all users from the database)

User ID	First Name	Last Name	Email	City	Edit	Delete
1	Mimi	Pham	mimi.pham@mail.com	San Antonio	Edit	Delete
2	Firstname 2	Lastname 2	Firstname2.Lastname2@mail.com	San Antonio	Edit	Delete
3	Firstname 3	Lastname 3	Firstname3.Lastname3@mail.com	San Antonio	Edit	Delete
4	Firstname 4	Lastname 4	Firstname4.Lastname4@mail.com	San Antonio	Edit	Delete
5	Firstname 5	Lastname 5	Firstname5.Lastname5@mail.com	San Antonio	Edit	Delete
6	Diemi	Pham	diemi.pham@mail.com	Austin	Edit	Delete
7	Alex	Pham	alex.pham@mail.com	San Antonio	Edit	Delete

[Add New User](#)
[Back to Menu](#)

Login page

← → ↻ ⓘ localhost:8090/users-webhibernate/Login.jsp ☆ ABP 🔍 ⬇ ⚙

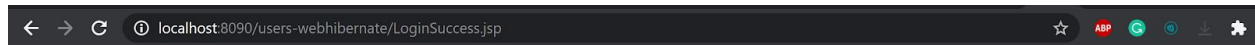
Login

Notes: username is your first name (case sensitive) and password is your email.

Username:

Password:

Login Success



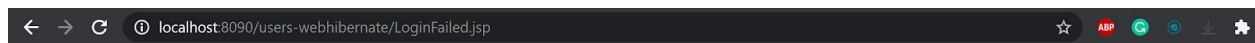
Welcome

Welcome Diemi

[Back to Menu](#)

[Logout](#)

Login Failed



Login Failed

Username or password is incorrect. Please try again.

Notes: username is your first name (case insensitive) and password is your email.

Username:

Password:

[Back to Menu](#)