COMP 3111: Software Engineering

Lab Activity: Web Site Security in ASP.NET
Add Web Site Security to the *HKeInvest* Web Application

If you have not attended the tutorial or read the tutorial and lab notes for *Web Site Security in ASP.NET*, you may not know how to complete this lab activity. Attending the tutorial will help ensure that you are able to complete the lab activity during the lab period.

LAB OBJECTIVE

Web site security is required to protect user information from non-authorized users as well as authorized users. In this lab activity you will add the capability for clients to create login accounts to access the client services of the ***HKeInvest*** web site and, in particular, to access only their own account information.

IMPORTANT: The following must be available for you to be able to complete this lab activity.

- The *HKeInvestDB.mdf* SQL Server database that you created in the "Managing Data Using SQL Server" lab activity containing the data that you added in that lab activity.
- The "SecurityHoldingDetails.aspx" web form that you created in the "Accessing a Database Using ADO.NET" lab activity.
- The "RegistrationPage.aspx" web form that you modified in the "Validating Data in ASP.NET" lab activity.

IMPORTANT: The ID properties of all controls should be set *exactly* as specified or you will get errors when you view the web form.

Download "HKeInvestIdentityManager-Download.zip" from the Tutorial and Lab Schedule section of the course web page. (***You must download the version for the lab activity, not the tutorial.***)

Download "HKeInvestData.cs" and drop it onto the Code_File folder in the `Solution Explorer`.

INSTALL IDENTITY MANAGER

Install *Identity Manager* in your web application according to the instructions in Appendix A of the "Web Site Security in ASP.NET" Tutorial and Lab Notes.

Note: *You must use the lab activity version of Identity Manager, not the tutorial version.*

CREATE ROLES AND A USER

According to the requirements given in the problem statement, the ***HKeInvest*** web site will require two roles Client and Employee. Therefore, create these two roles in your web application using *Identity Manager*.

Moreover, there will need to be at least one user in the Role "Employee". Therefore, add a user with user name "employee" and password "employee##" and assign the "Employee" role to this user.

Note: *You must have this user in your web application when you submit it for Activity 1.*

CUSTOMIZE USER NAME AND PASSWORD POLICIES

According to the problem statement, a password should be between 8 and 15 characters and should contain at least two non-alphanumeric characters. In the `Solution Explorer`, expand the `App_Start` folder and open the `IdentityConfig.cs` file. Find the line "RequiredLength = 6," and change it to "RequiredLength = 8," (*see page 3 of the notes*).

MODIFY THE ACCOUNT/LOGIN.ASPX WEB FORM

By default, the "Login.aspx" web form requires an email and password to log in. To change it to require a user name instead, change the appropriate properties of the "Email" TextBox, Label and RequiredFieldValidator controls to require a user name for login rather than an email address (*see pages 3 and 4 of the notes*).

CUSTOMIZE THE ACCOUNT/REGISTER.ASPX WEB FORM

To allow use of the existing Login controls in the "Account" folder and the ASP.NET authentication infrastructure, modify the "Account/Register.aspx" web form as follows (*see pages 4 and 5 of the notes*).

IMPORTANT: *Backup your HKeInvestWebApplication before proceeding to the following steps.*

CAUTION: Perform the following 4 steps with extreme care and double check what you are copying and deleting!

1. In the Source tab of the `Document` window for the "RegistrationPage.aspx" web form, select from the first `<div class="form-group">` tag up to the closing `</div>` tag located just above the `<div>` tag for the Button control and ***copy it***. ***Be careful not to include the Button control!***
2. Expand the `Account` folder in the `Solution Explorer` and open the "Register.aspx" web form.
3. In the Source tab of the `Document` window for the "Register.aspx" web form, select from the first `<div class="form-group">` tag up to the closing `</div>` tag located just above the `<div>` tag for the Button control and ***delete it***. ***Be careful not to delete the Button control!***
4. In the Source tab of the `Document` window for the "Register.aspx" web form, ***paste the text copied from the "RegistrationPage.aspx" web form*** at the location just above the `<div>` tag for the Button control.
5. In the "Register.aspx" web form, change the `<h4>` header text to "Create a new client account".
6. Set "EnableClientScript" to "False" for all Validation controls including the ValidationSummary control at the top of the web form.

(RE)CREATE CUSTOMVALIDATOR EVENT HANDLERS

Since you have copied the HTML source from the "RegistrationPage.aspx" web form to the "Register.aspx" web form, the event handlers for any CustomValidator controls that you created in the "RegistrationPage.aspx.cs" code-behind file do not exist in the "Register.aspx.cs" code-behind file and you need to recreate them. In particular, you need to recreate the "cvAccountNumber_ServerValidate" event handler and copy the code that you constructed in the "RegistrationPage.aspx.cs" code-behind file into the same event handler in the "Register.aspx.cs" code-behind file. If you have created any other CustomValidator controls in the "RegistrationPage.aspx.cs" code-behind file, then you also need to recreate these event handlers and copy their code from the "RegistrationPage.aspx.cs" code-behind file into the same event handler in the "Register.aspx.cs" code-behind file.

CUSTOMIZE THE ACCOUNT/REGISTER.ASPX.CS CODE-BEHIND FILE

By default, the user name is set to the email when a new account is created. Therefore, change the text "UserName = Email.Text" to "UserName = UserName.Text" on the third line of the "CreateUser_Click" event handler (*see page 6 of the notes*).

INITIAL TEST OF ACCOUNT/REGISTER.ASPX WEB FORM

Before You Test the Account/Register.aspx Web Form

As the "Account/Register.aspx" web form makes use of what you have done in several previous labs, you should go through the following checklist ***before*** you test it. If errors arise, go through this checklist again to make sure that your web forms pass the checklist.

- ☐ Check that you have pasted the correct code from the "RegistrationPage.aspx" web form into the "Register.aspx" web form.
- ☐ Check that you have set the EnableClientScript property to "False" for **all** validation controls.
- ☐ Check that you have pasted the correct code into the correct event handler in the code-behind file.
- ☐ Check that you have correctly customized the "Account/Register.aspx.cs" code-behind file.

Test the "Account/Register.aspx" web form with any valid data to check that it correctly creates a new user. If a new user is correctly created, then you should be redirected to the default web page for the web application and the user name you used should be displayed at the right side of the web page's navigation bar (*see Figure 11(b) of the notes*).

If errors occur, then do the following.

1. Go over the above checklist again!
2. Ask a TA for help.

COMPLETE THE CUSTOMIZATION OF THE ACCOUNT/REGISTER.ASPX.CS CODE-BEHIND FILE

So far you are able to create a user login account. However, you have not verified that the person creating the login account is the actual client that holds the account. You have also not assigned the user to the "Client" role or related the user name to the client information. Therefore, you need to modify the "Account/Register.aspx.cs" code-behind file to meet all of the above requirements.

Note: You can delete any user login accounts created in previous steps either using *Identity Manager* or directly in the *AspNetUsers* table, which can be accessed in the database found under the DefaultConnection in the `Server Explorer` window (*see Figure 7 in the notes*).

Verify Client Information

To verify that the person creating the account is the actual client that holds the account you need to check that *all* of the account and personal information entered into the "Register" web page actually matches the account and client information already stored in the database. If any of the information does not match, then you should abort creating the login account and inform the user about this situation.

Hint: You need to make this check *before* the user login account is created in the "CreateUser_Click" event handler and not create the user login account if the check fails.

Note: You can use the Literal control located at the top of the "Register.aspx" web form whose ID is "ErrorMessage" to display your error message (*see Figure 5 in the notes*).

Assign User to Client Role

You need to assign the user to the "Client" role after the user login account has been successfully created (*see Figure 5 of the notes for how to do this*).

Associate User Name to Client Information

You need to associate the client's user name with her account information. If login account creation is successful, then you can do this by updating the *userName* column in the *Account* table with the user name used to create the login account (*see Figure 5 in the notes and the "AddStudentRecord.txt" tutorial file*).

CREATE SECURE WEB PAGES

Both ***HKeInvest*** employees and clients will need to login to access certain information. To support this requirement, do the following.

1. Create two folders in the `Solution Explorer` and name them "ClientOnly" and "EmployeeOnly".
2. Add a `Web.config` file to each folder (*see page 8 of the notes*).
3. Set the access rules in the `Web.config` files so that all users are required to log in to access any file in the "EmployeeOny" and "ClientOnly" folders. Moreover, only users in the "Employee" role are allowed to access files in the "EmployeeOnly" folder and only users in the "Client" roles are allowed to access files in the "ClientOnly" folder (*see pages 9 and 10 of the notes*).
4. Drag the "SecurityHoldingDetails.aspx" web form into the "EmployeeOnly" folder.
5. Create a new web form named "ClientSecurityHoldingsDetails.aspx" in the "ClientOnly" folder and copy the HTML in the Source tab of the "SecurityHoldingDetails.aspx" web form into the Source tab of the "ClientSecurityHoldingsDetails.aspx".

   Note: ***Do not*** copy the "SecurityHoldingDetails.aspx" web form from the "EmployeeOnly" folder and paste it into the "ClientOnly" folder. Doing so will raise errors in Visual Studio when you try to run the web application due to naming conflicts.
6. Add the code from the "SecurityHoldingDetails.aspx.cs" code-behind file into the corresponding event handlers in the "ClientSecurityHoldingsDetails.aspx.cs" code-behind file.
7. Modify the code in the "ClientSecurityHoldingsDetails.aspx.cs" code-behind file in the "ClientOnly" folder so that it is not necessary to enter an account number to view the security holding of the logged in client to view the client's security holdings.

Note: You can use the method `Context.User.Identity.GetUserName()` to get the user name of the currently logged in user. You will need to add a `using Microsoft.AspNet.Identity` directive to your code-behind file to use this method.

IF YOU HAVE PROBLEMS LOGGING IN

- ☐ Check that the access rules have been set correctly for the "EmployeeOnly" and "ClientOnly" folders. In particular, check that *the rules are in the correct order*.
- ☐ Check that your user name has been assigned to the "Client" role. If it has not, you have missed adding the code that assigns a new user to the "Client" role in the "CreateUser_Click" event handler.

HOW TO GET THE CREDIT FOR THIS LAB

To get the full credit for this lab, you must do the following.

1. Create a client login account using your UST email login id as the user name. Since a user name must be at least 6 characters, if your UST email login id is less than 6 characters, then you should append the letter "z" as many times as is required to your UST email login id so that the user name is 6 or more characters.
2. Log in with this account.
3. Submit, by the end of the lab period, a printout of the "Security Holding Details" page, displayed in a browser, showing your UST email login id in the LoginName control, your student id, email, first name and last name in the Member Information page, and your name and student number in the page footer as shown in Figure 1.

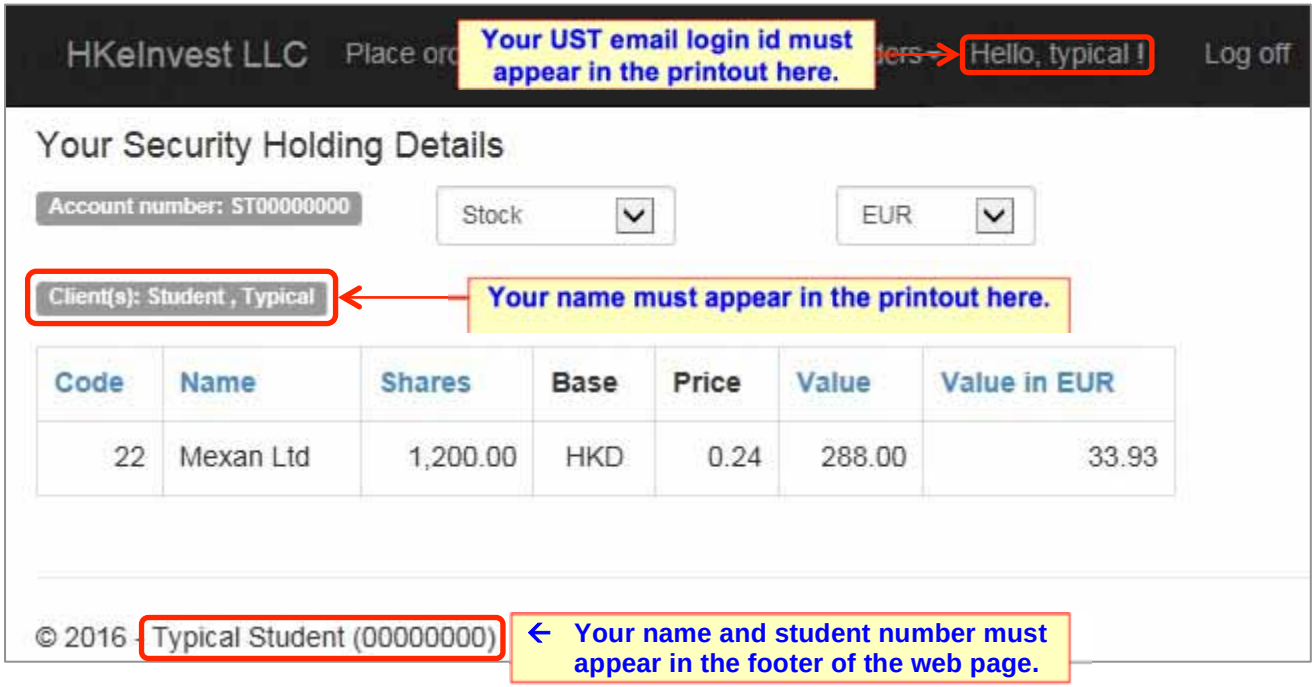

Figure 1: Example printout of Member Information page.

AFTER THE LAB

Add appropriate validation controls to the Account Application web form that you constructed after last week's lab activity, make the page functional (i.e., able to actually save client information in your database) and complete any requirements of previous lab activities that you have not completed. When you complete all the requirements of the first five labs, you will have completed all the Activity 1 requirements. ***Congratulations!***

IMPORTANT NOTE: ***The web application that you submit for Activity 1 must run on the computers in Lab 4.*** This is the computing environment that will be used to test your Activity 1 submission. Should your web application fail to run on the computers in Lab 4 when we test it, you are sure to lose marks.