COMP 3111: Software Engineering

Lab Activity: Accessing a Database Using ADO.NET
Implement Security Holding Details Page for *HKeInvest* Application

If you have not attended the tutorial or read the tutorial and lab notes for *Accessing a Database in ASP.NET using ADO.NET* you may not know how to complete this lab activity. Attending the tutorial will ensure that you are able to complete the lab activity during the lab period.

LAB OBJECTIVE

The ***HKeInvest*** application requires that an ***HKeInvest*** employee be able to request a listing of the details for a given security type in a specified account identified by the account number (see requirement 6 report (b)). In this lab activity you will create the web page for this requirement.

IMPORTANT: The *HKeInvestDB.mdf* SQL Server database that you created in last week's lab activity must be available in the App_Data folder of your "HKeInvestWebApplication" for you to be able to complete this lab activity. Moreover, it must be defined exactly as specified in last week's lab activity and contain all the records that you added, including the record containing your name and student number.

IMPORTANT: **DO NOT** add an SqlDataSource control to the web form that you will create in this lab activity.

Download the folder "ADONETLabActivity-Download.zip" from the Tutorial and Lab Schedule section of the course web page to the desktop and unzip it.

CONSTRUCT THE SECURITY HOLDING DETAILS WEB FORM

1. Add a new web form, name it "SecurityHoldingDetails.aspx" and inherit its layout from "Site.Master".
2. On the first line of the MainContent(Custom) ContentPlaceHolder add the text "Security Holding Details" and set its HTML tag to "<h2> Heading 2" (see Figure 1).
3. From the HTML palette of the `Toolbox`, add one `<div>` tag and three more `<div>` tags inside the top-level `<div>` tag.
4. Inside each of the three inner `<div>` tags add the controls shown in Figure 1 and set their properties as shown in Table 1.

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

MainContent (Custom)

Security Holding Details

Account number: Security type Currency

[lblClientName][lblResultMessage]

Code	Name	Shares	Base	Price	Value	Value in
Databound	Databound	Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound	Databound	Databound

Figure 1: Layout of the SecurityHoldingDetails web form.

Table 1: Properties of the controls in the SecurityHoldingDetails.aspx web form.

inner element	Control	Property	Value
1	Label	ID	
		AssociatedControlID	txtAccountNumber
		Text	Account number:
	TextBox	ID	txtAccountNumber
	DropDownList	ID	ddlSecurityType
		AutoPostBack	True
	DropDownList	ID	ddlCurrency
		AutoPostBack	True
		Visible	False
2	Label	ID	lblClientName
		Text	
		Visible	False
	Label	ID	lblResultMessage
		Text	
		Visible	False
3	GridView	ID	gvSecurityHolding
		Visible	False

Configure the Security Type DropDownList Control

1. Select "Edit Items…" in the "ddlSecurityType" DropDownList control's smart tag menu and add the following four Members in the `ListItem Collection Editor` window. ***Be careful to set the Text and Value properties correctly!***

Member	Text property	Value property
Security type	Security type	0
Bond	Bond	bond
Stock	Stock	stock
Unit Trust	Unit Trust	unit trust

2. In the "ddlSecurityType" DropDownList control's `Properties` window, select the Events tab and double-click the "SelectedIndexChanged" event. You will add code for this event later in the lab.

Configure the Currency DropDownList control

1. Select "Edit Items…" in the "ddlCurrency" DropDownList control's smart tag menu and add the following Member in the `ListItem Collection Editor` window.

Member	Text property	Value property
Currency	Currency	0

2. In the "ddlCurrency" DropDownList control's `Properties` window, select the Events tab and double-click the "SelectedIndexChanged" event. You will add code for this event later in the lab.

Configure the GridView Control

1. Select "Edit Columns…" in the "gvSecurityHolding" GridView control's smart tag menu.
2. In the `Fields` window, do the following.
   a. Uncheck the "Auto-generate fields" checkbox.
   b. Select "BoundField" in the "Available fields" pane, add seven fields and set their properties in the "BoundField properties" pane as shown in Table 2. The *DataField* property specifies the names of the columns in the *SecurityHolding* table while the *DataFormatString* property specifies how the data in the field should be formatted. The format {0:n2} specifies that a number should have two decimal places with commas inserted for thousands, millions, etc. as shown in Figure 2.

Table 2: Properties of "gvSecurityHolding" GridView control's BoundFields.

Field	Property				
	HeaderText	**ReadOnly**	**SortExpression**	**DataField**	**DataFormatString**
1	Code	True	code	code	
2	Name	True	name	name	
3	Shares	True	shares	shares	{0:n2}
4	Base	True		base	
5	Price	True		price	{0:n2}
6	Value	True	value	value	{0:n2}
7	Value in	True	convertedValue	convertedValue	{0:n2}

Note: Set only these properties. Do not change or set the value of any other property.

3. In the "gvSecurityHolding" GridView control's `Properties` window, select the Events tab and double-click the "Sorting" event. You will add code for this event later in the lab.

Add Databases and Code

1. From the "ADONETLabActivity-Download" folder drop the following items onto the indicated node in the `Solution Explorer`.
   a. Drop the "Code_File" folder onto the "HKeInvestWebApplication" node – This folder contains the classes
      - *HKeInvestData*, which provides the common code used to execute SQL statements on the *HKeInvestDB.mdf* SQL Server database that you created in last week's lab. ***You are not allowed to modify or add to the methods in this class.***
      - *HKeInvestCode*, which contains methods that are used in several other methods. ***You can add additional methods to this class.***
   b. Drop the "ExternalSystems" folder onto the "HKeInvestWebApplication" node – This folder simulates the functions provided by the external systems that manage securities and currency conversion, and that execute orders for securities. Among other things, it contains the class *ExternalFunctions*, which provides the methods necessary to access security and currency information and to submit orders for execution. (See the Course Project section of the course web page for details of these methods.) ***You are not allowed to modify or add to the methods in this class. You are allowed to directly use only the methods in the class ExternalFunctions.***
   c. Drop the SQL Server database *ExternalDatabases.mdf* onto the "App_Data" node – This database contains six tables—Bond, CurrencyRate, Order, Stock, Transaction and UnitTrust, which contain data about, respectively, bonds, currency rates, orders waiting to be executed, stocks, order transactions and unit trusts. (See the Course Project section of the course web page for details about the data contained in these databases.) ***You are not allowed to modify the descriptions of these tables or to access them directly. They can only be accessed using the methods in the ExternalFunctions class.***
   d. Drop the folder "ExternalSystemsTheme" onto the "App_Themes" node – This folder contains the Bootstrap Css stylesheet used as the theme for the external systems web pages. ***You are not allowed to modify this theme.***
2. At the top of the `Solution Explorer` window, select the "Show All Files"  tab and, for the folders "Code_File" and "ExternalSystems" do the following.
   a. Right-click the folder and select "Exclude From Project" from the popup menu.
   b. Right-click the folder again and select "Include In Project" from the popup menu.

   This makes all the files and folders within these two folders accessible to the web application.
3. Open the "Web.config" file in the Solution Explorer and add the code from the "AddTo-Webconfig.txt" file inside the `<connectionStrings>` tag. This code contains the connection string required to connect to the *HKeInvestDB.mdf* and *ExternalDatabases.mdf* SQL Server database.
4. In the "SecurityHoldingDetails.aspx.cs" code-behind file, do the following.
   a. Add the following three lines of code after the last "`using …`" statement at the top of the file. These statements allow access to the methods in these namespaces.

```
using System.Data;
using HKeInvestWebApplication.Code_File;
using HKeInvestWebApplication.ExternalSystems.Code_File;
```

b. After the first "{" in the partial class "SecurityHoldingDetails", add the following three lines of code. These statements create instances of the "HKeInvestData" "HKeInvestCode" and "ExternalFunctions" classes, which are needed to use the methods in these classes.

```
HKeInvestData myHKeInvestData = new HKeInvestData();
HKeInvestCode myHKeInvestCode = new HKeInvestCode();
ExternalFunctions myExternalFunctions = new ExternalFunctions();
```

c. Add the code in the following files into the corresponding event handlers. To create the last three event handlers, double-click the corresponding event in the Events tab of the control's `Property` window.

txt file	Event Handler
Page_Load.txt	Page_Load
ddlSecurityType_SelectedIndexChanged.txt	ddlSecurityType_SelectedIndexChanged
ddlCurrency_SelectedIndexChanged.txt	ddlCurrency_SelectedIndexChanged
gvSecurityHolding_Sorting.txt	gvSecurityHolding_Sorting

- The "Page_Load" event handler executes whenever a page is loaded in the browser. In this case the code in the event handler retrieves the available currencies from the external system and populates the "ddlCurrency" DropDownList control with the retrieved values *the first time the page is loaded*. The currencies are not retrieved again should the page be reloaded.
- The "ddlSecurityType_SelectedIndexChanged" event handler executes whenever the current item in the "ddlSecurityType" DropDownList changes. The code retrieves and displays the security holdings for a specified account. ***You need to complete this code during the lab (see below).***
- The "ddlCurrency_SelectedIndexChanged" event handler executes whenever the current item in the "ddlCurrency" DropDownList changes. The code converts and displays the value of a security holding in the selected currency. ***You need to complete this code after the lab.***
- The "gvSecurityHolding_Sorting" event handler executes whenever a column is sorted in the GridView. Because the GridView is populated programmatically, it is necessary to provide the code to implement the sorting.

5. Right-click the "HKeInvestWebApplication" node in the Solution Explorer and select "Add→Add Reference…" from the popup menu. In the `Reference Manager` window, find the entry "System.Windows.Forms", check the checkbox to its left and click the "OK" button. This includes the methods in this namespace in the web application. These methods are used to display error messages when executing SQL statements.

COMPLETE THE "DDLSECURITYTYPE_SELECTEDINDEXCHANGED" EVENT HANDLER CODE

The "ddlSecurityType_SelectedIndexChanged" event handler code has three TODOs for you to complete.

1. The first TODO requires you to set the string variables "accountNumber" and "securityType" from the values held by controls on the web form.
2. The second TODO requires you to construct the SQL statement that retrieves the client name(s).
3. The third TODO requires you to construct the SQL statement that retrieves the security holding of the client. For this SQL statement, the column values to retrieve and their order are specified by the *DataField* property of the first four BoundFields of the GridView control (see Table 2). Moreover, besides retrieving values for these four columns, the SQL statement also needs to return fixed values set to 0.00 for the remaining three columns of the GridView. This can be specified in an SQL Select statement as "0.00 as [columnName]" where "columnName" is replaced by the name of the DataField in the GridView control. The actual values of these last three columns are set by subsequent code. For example, the value of the *price* BoundField is obtained by code in the "ddlSecurityType_SelectedIndexChanged" event handler from the external system that manages securities using a method in the *ExternalFunctions* class. The value of the *value* BoundField is calculated by multiplying the number of shares held and the current price per share. Finally, the value of the *convertedValue* BoundField is calculated based on which currency is selected for conversion. You need to implement the currency conversion functionality after the lab.

For the SQL statements, you are highly advised to first test and debug them in the Query Editor before testing them in the web form.

Try to complete the TODOs on your own, but ask the TAs for help, if you are stuck.

HOW TO GET THE CREDIT FOR THIS LAB

Part 1: During the Lab session

To get half the credit for this lab you must submit, by the end of the lab period, a printout of the Security Holding Details web page, displayed in a browser, showing the securities held in your account, which you created in last week's lab. An example of such a printout is shown in Figure 2.

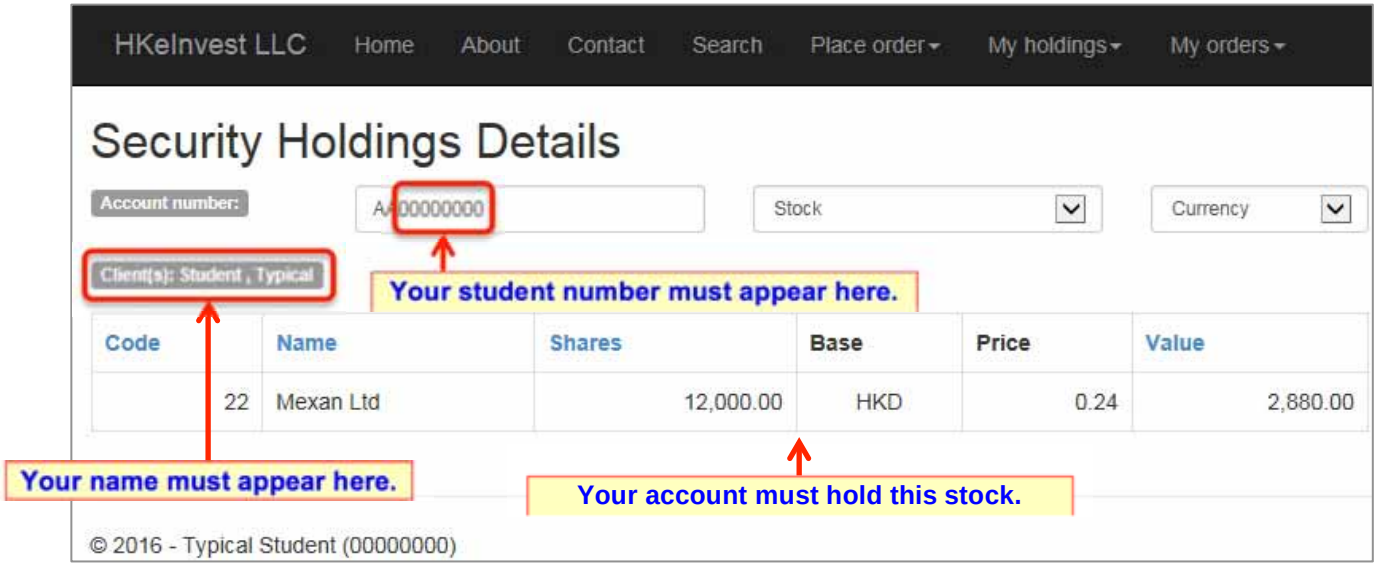

Figure 2: Example printout of Security Holding Details page.

Part 2: After the Lab Session

To get the remaining credit for this lab you need to complete the implementation of the currency conversion functionality in the "ddlCurrency_SelectedIndexChanged " event handler and demonstrate it in the lab next week.

Hint: You should consider using ViewState to cache the available currencies and their conversion rates when the page is first loaded so as to minimize how many times you need to access the external database.