# Unit 6 - Exercises

This exercise will requires additional files. Download the files from [here](https://thethirstycoder.gitlab.io/adev-2005-learn/unit_6/downloads/adev-2005_unit_6_exercise_files.zip).

The provided .zip file contains a Visual Studio Solution with one Windows Forms Application project. The project contains Form classes already designed for you. The Form classes may include behaviours already coded for you. You will edit the Form classes as prescribed in the exercises below.

## Data Binding

**EX1**. Setup a Simple Bind.

1. Open the SimpleBindForm.cs file in Code View.
2. Evaluate the code within the class. This exercise will bind data from a Student object to the UI controls. A Student has the following properties: FirstName, LastName and IsEnrolled.
3. Add implementation to the BindControls() method.
   1. Construct three Binding objects, one for each of the properties of the Student object referenced by the variable \_student.
   2. Add the Binding objects to the UI control’s DataBindings property.
4. Test the application. Choose “Simple Bind” from the main menu. You should see the UI controls reflecting the data of the Student object.
5. Click the View button and note the results.
6. Update the state of any of the controls.
7. Click the View button and note the results.

**EX2**. Setup a Complex Bind.

1. Open the ComplexBindForm.cs file in Code View.
2. Evaluate the code within the class. This exercise will bind data from a List (Collection) of Student objects to a UI control
3. Add implementation to the BindControls() method.
   1. Complex Bind the List to the dgvStudents DataGridView.
4. Test the application. Choose “Complex Bind” from the main menu. You should see the DataGridView reflecting the data of the Student objects in the List.
5. Click the View button and note the results. The dialog window displays data (state) from the first Student object in the List.
6. Double-click into any of the cells in the first row and edit the data.
7. Click the View button and note the results.

**EX3**. Setup a Complex Bind on a ComboBox.

1. Open the ComboBoxBindingForm.cs file in Code View.
2. Evaluate the code within the class. This exercise will bind data from a List of GroceryItem objects to a ComboBox.
3. Add implementation to the BindControls() method.
   1. Complex Bind the List to the ComboBox.
4. Test the application. Choose “ComboBox Bind” from the main menu. You should see the ComboBox reflecting the data of the GroceryItem objects in the List.
5. Click the View button and note the results. The dialog window displays data (state) from the selected item of the ComboBox.

### Questions

1. After completing the complex bind on the ComboBox, why are the items of the ComboBox displaying in that way?

**EX4**. Set how the ComboBox will represent the bound data.

Put the code for this exercise below the previous exercise code.

1. Set the ComboBox’s DisplayMember property to “Description” (the name of the property of the GroceryItem class).
2. Set the ComboBox’s ValueMember property to “UnitPrice” (the name of the property of the GroceryItem class).
3. Test the application. The ComboBox will bind as it did in the previous exercise.
4. Click the View button and note the results.

**EX5**. Use Simple and Complex binding to automate UI updating.

Put the code for this exercise below the previous exercise code.

1. Add code to Simple Bind the “UnitPrice” property of the GroceryItem objects to the lblUnitPrice Label.
2. Test the application. You should see the unit price state of the selected item display in the Label.

**EX6**. Define how bound data is formatted.

Put the additional code for this exercise below the previous exercise code.

1. Modify the Binding object from the previous exercise such that formatting is enabled.
2. Set the FormatString property of the Binding object to the currency format string.

**EX7**. Use BindingSource methods to navigate through bound data.

1. Open the BindingSourceForm.cs file in Code View.
2. Evaluate the code within the class. This exercise will bind data from a List of Student objects to a various controls.
3. Test the application. You should see the first Student in the List represented in the UI controls.
4. Add implementation to the Click event handler methods for the Next, Previous, First and Last Buttons. The implementation will invoke the appropriate BindingSource method to update the UI controls.
5. Test the application. You should see the UI controls update as you navigate through the data.

**EX8**. Explore the Current property of a BindingSource object.

1. Add implementation to the Current Button Click event handler. The implementation will display the value of the Current property of the BindingSource in a MessageBox.
2. Test the application. As you navigate through the data, click the Current Button and note the results.

**EX9**. Handle the PositionChanged event of a BindingSource object.

1. In the Load event handler of the Form, subscribe to the PositionChanged event of the \_bindingSource object reference.
2. Add implementation to the event handler such that it will update the lblRecord Label text to display the Position and Count of the BindingSource object in the format: {position + 1} of {count}. Example: 1 of 10.
3. Test the application. As you navigate through the data, the Label will indicate the record number you are currently viewing.